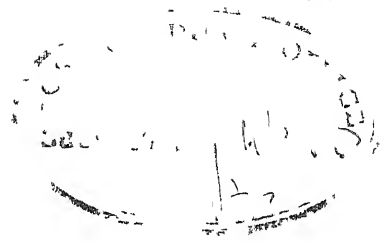
EFFECTIVE ALGORITHMS FOR THE CYCLIC STAFF SCHEDULING PROBLEM

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By
S. V. RAMA REDDY

to the

INDUSTRIAL & MANAGEMENT ENGINEERING PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
MARCH, 1985



CERTIFICATE

This is to certify that the work embodied in the thesis entitled, "Effective Algorithms for the Cyclic Staff Scheduling Problem," by S.V. Rama Reddy has been carried out under our supervision and has not been submitted elsewhere for the award of a degree.

(R. K. Ahuja)
Lecturer
Industrial and Management Engg.
Indian Institute of Technology
Kanpur 208016, INDIA

(J.L. Batra)
Professor and Head
Industrial and Management Engg.
Indian Institute of Technology,
Kanpur 208016, INDIA

March, 1985

ACKNOWLEDGEMENTS

I express my deep sense of gratitude to my thesis supervisors, Dr. R.K. Ahuja and Dr. J.L. Batra, for the responsible and stimulating guidance and profuse assistance I have received from them throughout the course. Indeed, they are responsible for taking out the best in me.

I am thankful to my friend, Mr. V.V.S.N. Murty, for his help in many ways. I am also thankful to Mr. S.S.R. Murty, Mr. M.S. Suryanarayana and other friends for their ready willingness to help me at any time.

I thank Mr. J.K. Misra for his excellent typing, Mr. Bajpai for his appreciable plotting of charts, and Mr. Buddhi Ram Kandiyal for his immaculate cyclostyling work.

S.V. Rama Reddy

CONTENTS

CHAPTER I

INTRODUCTION

1.1 Introduction:

In recent years, much interest has been focussed on various problems that arise in cyclic staffing. Such problems involve the optimal scheduling of resources to meet demands where both resource availability and demand profile are cyclic.

Many industries and public services operate around the clock, 7 days a week. This is the case in police and fire departments, in telephone companies, in hospitals and in many heavy industries. Typical work schedules for employees of such services and industries normally consist of a cycle spreading over a few weeks and in which shift work and holiday periods alternate. These schedules consist of a cycle which repeats itself after a few weeks. Employees can therefore be seen as continuously going through the same basic work-holiday pattern or rotating schedule. One rotating schedule is determined by the length of its cycle, by the length and frequency of the work and rest periods and by the way these alternate.

The objective in cyclic staff scheduling problem is to minimize the total labor hours needed to meet demand requirements during different time periods. This problem is efficiently solvable when resources are continuously available during consecutive time periods. For an example of continuous availability,

let us consider one fundamental problem, called (5-7) cyclic staffing problem. In this problem, both resource availability and requirements are cyclic. The objective is to minimize the number of workers needed to meet daily requirements, with the added provision that each worker be allowed two consecutive days off each week.

Let x_j denote the number of workers in shift j and b_i denote the requirement during time period i. With these notations the above problem can be formulated as:

$$\text{Minimize} \quad \sum_{j=1}^{7} x_j$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \geq \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$$

$$x_j \geq 0 \text{ and integer for all } j = 1, \ldots, 7.$$

Here the work-shift is five consecutive days. If we denote the 0-1 matrix by A, each column of A represents a possible work shift and each row of A shows the shifts that are active on the corresponding day. That the resources are continuously available is reflected in A, where each column contains exactly

one block of consecutige ones (when viewed circularily). This problem has been solved optimally by Bartholdi et al [6].

In a more realistic model, resources are intermittently available. This may be due to, for example, lunch and relief breaks for the workers. The following matrix is an hypothetical example showing intermittently available resources.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

This is an (8 x 8) matrix and shift length is five. It can be observed that there is one break (when viewed circularity), i.e. two blocks of consecutive ones, in each row.

Bartholdi [5] showed that the problem of cyclic staff scheduling with intermittently available resources is a NP-complete problem. In view of this observation, the possibility of developing an exact algorithm for the above problem is remote. Heuristic approaches based on linear programming relaxation, network relaxation or lagrangean relaxation are available in literature. In all of them, some explicit use was made of the fact that A almost had the property of consecutive ones.

Bartholdi [5] developed an algorithm based on linear programming round-off by making no assumptions about the circularity of A. He gives a bound on the maximum possible deviation of his heuristic from the integer optimum solution. This bound is not good for the problems in which intermittent breaks are reasonably large in each shift. More often than not, it is observed that the deviation of the solution obtained using Bartholdi's method from the integer optimum solution is almost equal to the bound given by him.

In this dissertation, we consider the problem of scheduling with intermittently available resources and develop some methods for solving it. We develop three algorithms. One is LP-based heuristic algorithm. In this algorithm, a sequence of LP's are solved and an improved version of the rounding-off procedure suggested by Bartholdi et al [6] is used after solving each LP to get a feasible solution to the cyclic staff scheduling problem. The other two algorithms are Branch and Bound algorithm and cutting plane algorithm. These two are exact algorithms. Branch and Bound algorithm makes use of sensitivity analysis to impose lower or upper bound constraints on any variable and LP-based heuristic is applied at every node in the branching tree. The cutting plane algorithm makes use of the dual fractional cutting plane algorithm. Two types of cuts are introduced to reduce the total number of cuts needed.

All the suggested algorithms have been programmed and tested extensively on randomly generated cyclic staff scheduling problems. The results are quite encouraging and reported. The solutions obtained using LP-based heuristic algorithm are observed to be very close to the optimum solution for all the problems solved. The Branch and Bound algorithm is able to solve problems of size (96 x 96) optimally in a few minutes of CPU time. The performance of cutting plane algorithm was observed to be erratic. It could not solve problems of size more than (48 x 48).

1.2 Preliminaries:

Some notations and well-known concepts of cyclic staff scheduling problem are used throughout the thesis. For the sake of completeness they are given below.

A 0-1 vector is said to be circular if its 1's occur consecutively, where the first and last entries are considered to be consecutive.

A matrix is called column circular (resp., row circular) if its columns (resp., rows) are circular.

The term scheduling in this context, denotes the act of specifying how many resources should there be in different shifts so as to satisfy the requirements during different time periods with minimum cost.

Following notations are used in the thesis:

m total number of time intervals to be scheduled.

n total number of different shifts or work schedules.

A an $(m \times n)$ matrix of 0's and 1's.

x_j number of operators assigned to the j-th work schedule.

b_i number of operators required for time interval i.

$a_{ij} = \begin{cases} 1 & \text{if time interval } i \text{ is a work period in the j-th work schedule.} \\ 0 & \text{otherwise.} \end{cases}$

c_j cost of assigning a worker to shift j.

q maximum number of blocks of consecutive ones (when considered circularity) in any row in matrix A.

k minimum number of ones in each column of A.

X assignment vector $(n \times 1)$.

C cost vector $(1 \times n)$.

b requirement vector $(m \times 1)$.

0 vector of zeros $(m \times 1)$.

1 vector of ones $(1 \times n)$.

$\lfloor . \rfloor$ the greatest integer less than or equal to '.' .

$\lceil . \rceil$ the least integer greater than or equal to '.' .

1.3 Statement of the Problem.

The problem of cyclic staff scheduling can be posed as follows: given resource requirements for each time period within

a planning horizon, find out a schedule of resources to meet these requirements at minimum cost. This can be formulated as an ILP as given below.

$$\text{Min.} \quad \sum_{j=1}^{n} c_j x_j$$

s.t.

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \geq b_i, \quad i = 1, \ldots, m$$

$$x_j \geq 0, \quad j = 1, \ldots, n$$

and integer.

This can alternatively be stated as,

$$\text{Min. } C\,X$$

s.t.

$$A\,X \geq b$$

$$X \geq 0 \text{ and integer,}$$

where C, b are integral vectors and A is an m x n matrix with all entries either zero or one. Without loss of generality we assume that $b > 0$, $C \geq 0$, and that A has atleast one 1 in each row. Each b_i may be imagined to represent the number of workers required during time period i, and each c_j the cost of assigning a worker to shift j. Each column of A may be imagined to represent a possible work shift.

In this dissertation, we consider a special case of the above problem that is $c_j = 1, \ \forall\ j = 1, \ldots, n$. The problem then is to determine smallest number of workers to satisfy the demand schedule.

Mathematically the problem is.

Minimize $1X$

s.t.

$$A X \geq b$$

$$X > 0 \text{ and integer.}$$

Subsequently this problem is referred to as cyclic staff scheduling (CSS) problem.

1.4 Literature Review:

The cyclic staff scheduling problem has been studied by several researchers in the literature. Dantzig [7] formulated the general CSS problem as an ILP as given below.

$$\text{Min.} \quad \sum_{j=1}^{n} c_j X_j$$

s.t.

$$\sum_{j=1}^{n} a_{ij} X_j \geq b_i, \quad i = 1, \ldots, m$$

$$X_j \geq 0, \text{ integer}, \quad j = 1, \ldots, n.$$

where X_j is the number of workers in shift j, and C_j is the cost of assigning a worker to shift j, $a_{ij} = 1$ if shift j is present on i-th time period and $a_{ij} = 0$ otherwise. The b_i is the requirement during time period i.

Henderson [9] has considered a special case of Dantzig's general ILP formulation for scheduling of telephone operators as given below.

$$\text{Min.} \quad \sum_{j=1}^{n} X_j$$

s.t.

$$\sum_{j=1}^{n} a_{ij} X_j \geq b_i, \quad i = 1, \ldots, m$$

$$X_j \geq 0, \text{ integer}, \quad j = 1, \ldots, n$$

Luce [12] has suggested an alternative approach for scheduling of telephone operators. He essentially modified the ILP formulation of the CSS problem suggested by Henderson by adding shortage and surplus tolerance variables. Luce's ILP problem is more difficult than solving Henderson's problem as more integer variables are needed for formulation. However, a constraint fixing the total number of operators can be added in Luce's model. The underlying assumption of Luce's model is that "cost" of deficit equals "cost" of surplus. Penalizing under-staffing and over-staffing equally is indeed a little peculiar and unrealistic.

The cyclic staff scheduling problem is more easily solvable when resources are continuously available. Baker [3] has suggested a method which needs only hand computations to solve the (5-7) cyclic staffing problem optimally. In this problem, full-time workers are available for five consecutive days in a one week cycle with two consecutive days off.

Baker [4] has suggested another simple algorithm for the problem of allocating staff in order to meet demand over a

repeating seven-day cycle when both full time and part-time workers are there. The formulation assumes that full-time employees are entitled to two consecutive days off and that the objective is to meet demand for staff with a minimumm use of part-time employees.

For solving the general cyclic staffing problem with continuously available resources, Bartholdi et al [6] suggested an optimal technique based on linear programming round-off. In this method, the given CSS problem is first solved as an LP by relaxing the integrality restriction on all the variables. The continuous valued optimum solution obtained is then rounded-off in a special way, which guarantees the feasibility and optimality of the rounded-off solution to the original problem.

In practice, resources may not be available continuously. This may be due to lunch or relief breaks for the workers. The CSS problem with intermittently available resources is more difficult to solve. Current applications resolve the dilemma of problem size in a variety of ways. One possibility is to solve the mCSS problem as an LP and then to use some heuristic procedure to round-off the resulting non-integer solution.

The scheduling of intermittently available resources has been studied by Glover and Mulvey [8], Segal [18], Henderson and Berry [10], Laporte [11], Shepardson and Marsten [19], Bartholdi [5], Ondhensden and Wen-Jang [16], Willis and Wilson [21] Mabert and Watts [13], and Morris and Showalter [14].

Segal [18] considered the problem of scheduling telephone operators where operators work in shifts that are specified by start and end times. Each tour contains from one to three relief periods during which the operators are not working, and a particular schedule of relief periods within a tour is called a trick. He proposes a method for determining the assignment of operators to each trick, it utilizes the out-of-Kilter algorithm and two network-flow models. The problem is formulated as an ILP.

$$\text{Min.} \quad \sum_{j=1}^{n} C_j x_j$$

s.t.

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad \forall \ i = 1, \ldots, m$$

$$x_j \geq 0, \text{ integer}, \quad \forall \ j = 1, \ldots, n.$$

He considers this problem unsolvable with the available integer programming techniques at that time. He follows an approach based on network-flow for formulation in which integer solutions are guaranteed. Initially breaks and relief periods are ignored and the simplified problem is solved using out of Kilter algorithm. He then brings in relief and break periods and uses some heuristic procedure to meet shortage at break and relief periods.

Henderson and Berry [10] gave some heuristic methods for scheduling of telephone traffic exchange operators to meet demand that vary over a 24-hour operating period. Two types of heuristics are described:

(1) for determining the work shift types to be considered in preparing an operator shift schedule and,

(2) for constructing an operator shift schedule from a given set of work shift types.

They give two heuristics for determining the set of shift types. Once the working subset (shifts) has been defined, the given CSS problem is solved as a relaxed LP and all the variables are rounded-up and then two heuristics are applied to remove the excess operators.

Bartholdi [5] has suggested a guaranteed accuracy round-off algorithm for solving the CSS problem with intermittently available resources. We describe his method somewhat in detail in a separate section.

Oudheusden and Wen-Jenq [16] consider the following problem, given operator requirements during different time periods of the day, a certain number of operators are assigned to the particular work schedules in order to meet, as well as possible, the varying demand for operators. If the number of operators for a day are fixed at p, the following model is suggested,

Max. Y

s.t.

$$\left(\sum_{j=1}^{n} a_{ij} X_j - b_i \right) \geq Y, \quad i = 1, \ldots, m$$

$$\sum_{j=1}^{n} X_j = p$$

$$X_j \geq 0, \text{ integer}, \quad j = 1, \ldots, n.$$

Y is an integer variable which can be positive, zero or negative.

This is a maxmin problem as it maximizes the smallest difference between supply and demand in any period. When Y^{opt} (optimum objective function value), takes the value -1, at least one time interval will be understaffed by one operator. When Y^{opt} equals 0, all demands are met and overstaffing in all periods is not possible. When Y^{opt} equals + 1, all operator requirements are met with atleast one extra worker. Results of one realistic study, applied to Taiwan Telecommunication administrative Bureau are presented.

Arthur and Ravindran [2] consider a problem of scheduling nurses under multiple objectives. They make use of a goal programming model which allows for consideration of multiple conflicting objectives inherent in scheduling a nursing staff.

Willis and Wilson [21] present a case study of scheduling telephone betting operators. They demonstrate the application of linear programming and network flow techniques for determining optimal shift sizes for telephone betting operators. Mabert and Watts [13] present a simulation analysis of tour shift construction procedures. They provide some managerial suggestions on the issues of effectively staffing personnel on a weekly basis.

Morris and Showalter [14] give an exact and a heuristic algorithms for solving the CSS problem with intermittently available resources. First they consider the problem of

scheduling workers on a day, where the workers work for 8 hours with a one hour break after 4 hours of work in 24-hour cycle. For solving this (24 x 24) problem optimally, they suggest a mixed cutting plane and branch and bound algorithm. They give some heuristic approaches for solving the weekly scheduling problem of (168 x 168) size.

1.4.1 Bartholdi's Method.

Bartholdi [5] has suggested a heuristic algorithm to find a feasible solution for the CSS problem when there are intermittent breaks in the shift. In this section, we describe his method in detail.

Let A be an arbitrary m x n 0-1 matrix whose i-th row is $A_{i.}$ and whose j-th column is $A_{.j}$. Suppose each column of A has atleast K ones but no row has more than q blocks of consecutive ones (when considered circularity). A vector whose every entry is (q - 1) is denoted by (q-1).

The algorithmic steps involved in his method are given below.

Step 1: Solve the continuous-valued linear program LP(b+q-1):

$$\text{Min. } 1\,X$$

s.t.

$$A\,X \geq b + q - 1$$

$$X \geq 0$$

Let the solution be $x_{LP}\,(b+q-1) = (x_1, x_2, \ldots, x_n)$.

Step 2: Construct the (integer-valued) heuristic solution,

$$x_h = (\lceil x_1 \rceil, \lceil x_1+x_2 \rceil - \lceil x_1 \rceil, \ldots, \lceil \sum_{h=1}^{n} x_h \rceil - \lceil \sum_{h=1}^{n-1} x_h \rceil).$$

Bartholdi has proved that the solution obtained after rounding-off as described above is feasible to the following problem.

$$\text{Min} \quad 1\ X$$

s.t.

$$A\ X \geq b$$

$$X \geq 0, \text{ integer}$$

He also obtains an upper bound on the possible deviation of his heuristic solution (Z_h) from the integer optimum solution Z_{IP} to the staffing problem. This bound is,

$$Z_h - Z_{IP} \leq \left| \frac{n}{k} (q - 1) \right|.$$

It might be noted that this bound is not satisfactory for higher values of q.

1.5 Outline of the Thesis:

A brief chapter by chapter outline of the thesis is given in this section.

In Chapter II, we develop a heuristic algorithm for the CSS problem. In this method, the given staffing problem is solved as an ordinary LP by relaxing integrality restriction on all variables. An improved version of the special rounding-off procedure suggested by Bartholdi et al [6] is applied. If no

constraint is violated with the rounded-off solution, this solution is optimum. Otherwise, we find out the unsatisfied demand in each time period. This demand is taken as the requirement of the new CSS problem. This is repeated until there is no unsatisfied demand for every time period. The solution obtained using this algorithm is found to be very close to the optimum solution. A numerical example is presented to illustrate the algorithm.

In Chapter III, we develop an adaptation of the general integer programming branch and bound algorithm and a dual fractional cutting plane algorithm for the CSS problem. Some of the important features of this algorithm are.

(i) Only one A-matrix of fixed size is needed to be stored throughout the problem,

(ii) Reoptimization is greatly used to reduce the computations substantially,

(iii) To impose lower (upper) bound type of constraints on any variable, sensitivity analysis of the lower (upper) bound is performed,

(iv) The heuristic developed in Chapter II is applied at every node of the branch and bound tree to obtain a good feasible solution,

(v) The variable selected for branching is such that the pruning is more effective.

Detailed discussion of all these aspects is presented. A numerical example is presented to clarify the above steps.

We also develop a cutting plane algorithm for finding an optimum solution to the CSS problem. Dual fractional cutting plane algorithm is used. Two types of cuts are introduced. It is known that in an integer optimum solution to the CSS problem, sum of all variables must be an integer, so whenever the objective value is found to be fractional, we apply the cut that the sum of all original variables must be greater than or equal to the least integer greater than the objective function value. If the objective function value is fractional one, we then apply Gomory's fractional cut. Detailed discussion about the strategy used for finding source variable for cut generation is also given.

In Chapter IV, we present the computational performance of Branch and Bound algorithm, LP-based heuristic, cutting plane algorithm and Bartholdi's method. The results are found to be quite encouraging. The solutions obtained using LP-based heuristic were found to be very close to the optimum solution. The number of LPs solved in any problem were never more than 2. The Branch and Bound algorithm is able to solve problems of size (96 x 96) optimally in a few minutes of CPU time. The performance of cutting plane algorithm was observed to be erratic. It could not solve problems of size more than (48 x 48). Computer programs for all these algorithms were written in FORTRAN-IV and implemented on DEC-1090 Computer System.

CHAPTER II

LP - BASED HEURISTIC ALGORITHM

2.1 Introduction:

In this chapter, a new heuristic algorithm for solving the CSS problem is developed. The algorithm is called LP-based heuristic algorithm as a sequence of LP's are solved to get a good feasible solution to the CSS problem. We first describe the intuitive ideas that led to its development. A stepwise description of the algorithm is then presented. A numerical example is solved to illustrate the algorithm.

2.2 Development of the Heuristic Algorithm:

Bartholdi [5] developed a heuristic algorithm for solving the CSS problem. He gave a priori bound for the maximum possible deviation of his heuristic solution (Z_h) from the integer optimum solution (Z_{IP}) for the CSS problem. The bound is

$$Z_h - Z_{IP} \leq \left\lceil \frac{n}{k} (q - 1) \right\rceil .$$

His algorithm does not give a good solution for the CSS problem when there are large number of intermittent breaks (i.e., Bartholdi's q is large) and/or when n/k ratio is large. For example, if we consider a (96 x 96) problem with k = 30 and q = 6, Bartholdi's bound is $\left\lceil \frac{96}{30} (6-1) \right\rceil = 16$ units. Further, it was observed by us computationally that his algorithm

invariably requires as many additional units as given by his bound because he is adding (q-1) additional units to the actual demand in each time period in the initial stage to obtain a feasible solution to the original CSS problem. We observed that there is no need to add (q-1) units to the requirement in each time period in the initial stage itself without examining whether it is really needed or not. Our strategy is to provide the additional number of workers on those days where the need arises after applying the special rounding-off to the first LP solution obtained. With this intuitive idea in mind, we develop the LP-based heuristic algorithm for solving the CSS problem.

Familiarity with the following special way of rounding-off is needed before going through the formal description of the algorithm. Given n continuous valued variables say $x_1, x_2, \ldots, x_n$, they can be rounded-off in the following special way suggested by Bartholdi et al [6].

$$Y = (\lceil x_1 \rceil, \lceil x_1+x_2 \rceil - \lceil x_1 \rceil, \ldots, \lceil \sum_{j=1}^{n} x_j \rceil - \lceil \sum_{j=1}^{n-1} x_j \rceil) \qquad (2.2.1)$$

Bartholdi has shown that if a CSS problem is solved as an ordinary LP and the continuous n values obtained are rounded-off in the above fashion, there will not be more than (q-1) units of unsatisfied demand on any of the time periods. We generalize this result to the n integer solutions obtained by the following round-offs.

$$Y^1 = (\lceil x_1 \rceil, \lceil x_1+x_2 \rceil - \lceil x_1 \rceil, \lceil x_1+x_2+x_3 \rceil - \lceil x_1+x_2 \rceil, \ldots, \lceil \sum_{J=1}^{n} x_J \rceil - \lceil \sum_{J=1}^{n-1} x_J \rceil) \quad (2.2.1.1)$$

$$Y^2 = (\lceil \sum_{J=1}^{n} x_J \rceil - \lceil \sum_{J=2}^{n} x_J \rceil, \lceil x_2 \rceil, \lceil x_2+x_3 \rceil - \lceil x_2 \rceil, \ldots, \lceil \sum_{J=2}^{n} x_J \rceil - \lceil \sum_{J=2}^{n-1} x_J \rceil) \quad (2.2.1.2)$$

.

.

.

$$Y^n = (\lceil x_n+x_1 \rceil - \lceil x_n \rceil, \lceil x_n+x_1+x_2 \rceil - \lceil x_n+x_1 \rceil, \lceil x_n+x_1+x_2+x_3 \rceil - \lceil x_n+x_1+x_2 \rceil, \ldots, \lceil x_n \rceil) \quad (2.2.1.n)$$

where, $x_J = x_{J-n}$, if $J > n$.

To find-out whether a given solution is feasible to the given CSS problem or not we find out the unsatisfied demand for each time period with that solution. If the unsatisfied demand on all the time periods is zero, the given solution is feasible to the CSS problem. Let $Y^h = \{y_1^h, y_2^h, \ldots, y_n^h\}$ denote the rounded-off solution vector corresponding to the h-th rounded-off solution, where y_1^h denotes the rounded-off value of the variable x_1 in the h-th solution vector. Let $U^h = \{u_1^h, u_2^h, \ldots, u_m^h\}$

denote the vector of unsatisfied demands when Y^h is taken as the solution for the CSS problem. u_i^h denotes the unsatisfied demand on i-th time period when Y^h is taken as the solution for the CSS problem. Each u_i^h is obtained as given below:

$$u_i^h = \text{maximum } \{ 0, b_i - \sum_{j=1}^{n} a_{ij} \cdot y_j^h \} \qquad (2.2.2)$$

$$\forall\ i = 1, \ldots, m, \quad \forall\ h = 1, \ldots, n$$

Now, we explain how the LP-based heuristic proceeds to get a good feasible solution to the CSS problem. We first ignore the integrality restriction on all the variables and solve an LP to get continuous valued optimum solution to the CSS problem. Now, n-rounded-off solutions Y^1, Y^2, ..., Y^n and the corresponding unsatisfied demand vectors U^1, U^2, ..., U^n are obtained. Total unsatisfied demand for solution Y^h is given by $\sum_{i=1}^{m} u_i^h$. We calculate the total unsatisfied demand for each of the n solutions obtained and select the one which gives the minimum amount of total unsatisfied demand. Let this solution be Y^p. If the total unsatisfied demand for this solution is zero, we have a feasible solution to the CSS problem and stop. Otherwise, a new LP is formulated by taking the demand vector for the given CSS problem as U^p. We now solve this LP and obtain n rounded-off solutions and select the one with minimum amount of total unsatisfied demand. The values of all y_j^p's, of the best solution obtained here are added to the corresponding rounded-off y_j^p's

obtained upto the previous LP. This process is repeated until the total unsatisfied demand becomes zero.

A formal statement of the algorithm is given in the next section.

2.3 Statement of the LP-Based Heuristic Algorithm:

Step 0: Set $X^* = 0$.

Step 1: Let X be the solution of the following continuous valued variable CSS problem.

Min. $1\ X$

s.t.

$$A\ X \geq b$$

$$X \geq 0$$

Step 2: Obtain the n round-off integer solutions $Y^1, Y^2, \ldots, Y^n$ using (2.2.1.1) to (2.2.1.n). Compute the vectors of unsatisfied demands $U^1, U^2, \ldots, U^n$ using (2.2.2) corresponding to $Y^1, Y^2, \ldots, Y^n$ respectively. Let Y^p be the solution for which

$$\sum_{i=1}^{m} u_i^p \leq \sum_{i=1}^{m} u_i^j, \quad \forall\ j = 1, \ldots, n.$$

Set $X^* = X^* + Y^p$.

If $\sum_{i=1}^{m} u_i^p = 0$ go to Step 3, otherwise set $b = U^p$ and go to Step 1.

Step 3: The vector X^* is the heuristic optimum solution of the given CSS problem. STOP.

Observation.

It has been computationally observed that deleting all those constraints for which b_i is zero while solving the LP in Step 1 results in significant reduction in computational time.

2.4 Numerical Example:

We now solve a numerical example to illustrate various steps involved in the LP-based heuristic algorithm. Consider the following hypothetical CSS problem.

$$\text{Min.} \quad \sum_{j=1}^{12} x_j$$

s.t.

$$\begin{bmatrix} 1&0&0&0&0&1&0&1&1&0&1&1 \\ 1&1&0&0&0&0&1&0&1&1&0&1 \\ 1&1&1&0&0&0&0&1&0&1&1&0 \\ 0&1&1&1&0&0&0&0&1&0&1&1 \\ 1&0&1&1&1&0&0&0&0&1&0&1 \\ 1&1&0&1&1&1&0&0&0&0&1&0 \\ 0&1&1&0&1&1&1&0&0&0&0&1 \\ 1&0&1&1&0&1&1&1&0&0&0&0 \\ 0&1&0&1&1&0&1&1&1&0&0&0 \\ 0&0&1&0&1&1&0&1&1&1&0&0 \\ 0&0&0&1&0&1&1&0&1&1&1&0 \\ 0&0&0&0&1&0&1&1&0&1&1&1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} \geq \begin{bmatrix} 123 \\ 123 \\ 122 \\ 125 \\ 123 \\ 124 \\ 124 \\ 122 \\ 123 \\ 124 \\ 124 \\ 123 \end{bmatrix}$$

$x_j \geq 0$, integer, $\forall \; j = 1, \ldots, 12$.

Step 1: This problem is first solved by relaxing the integrality restriction on all x_j's as an ordinary LP. The optimal continuous valued variable vector is:

X = (39.80, 0.80, 40.60, 0.80, 41.40, 0.40, 40.40, 0.00, 41.60, 0.00, 40.80, 0.40)

Step 2: The n rounded-off solutions are:

Y^1 = (40, 1, 41, 0, 42, 0, 41, 0, 41, 0, 41, 0)
Y^2 = (39, 1, 41, 1, 41, 0, 41, 0, 41, 0, 41, 1)
Y^3 = (40, 0, 41, 1, 41, 1, 40, 0, 42, 0, 40, 1)
Y^4 = (40, 1, 40, 1, 42, 0, 40, 0, 42, 0, 41, 0)
Y^5 = (40, 1, 41, 0, 42, 0, 41, 0, 41, 0, 41, 0)
Y^6 = (40, 1, 40, 1, 41, 1, 40, 0, 42, 0, 41, 0)
Y^7 = (39, 1, 41, 1, 41, 0, 41, 0, 41, 0, 41, 1)
Y^8 = (40, 1, 40, 1, 42, 0, 40, 0, 42, 0, 41, 0)
Y^9 = (40, 1, 40, 1, 42, 0, 40, 0, 42, 0, 41, 0)
Y^{10} = (39, 1, 41, 1, 41, 0, 41, 0, 41, 0, 41, 1)
Y^{11} = (39, 1, 41, 1, 41, 0, 41, 0, 41, 0, 41, 1)
Y^{12} = (40, 0, 41, 1, 41, 1, 40, 0, 42, 0, 40, 1)

The unsatisfied demand vectors and the total unsatisfied demand for each of the above 12 solutions are given in Table 3.1.

day	1	2	3	4	5	6	7	8	9	10	11	12	T.U.S. Demand
U_1	1	0	0	1	0	0	0	0	0	0	1	0	3
U_2	1	0	0	0	0	1	0	0	0	1	0	0	3
U_3	0	0	1	0	0	1	0	0	0	0	0	1	3
U_4	0	0	0	0	0	0	1	1	0	0	0	0	2
U_5	1	0	0	1	0	0	0	0	0	0	1	0	3
U_6	0	0	0	0	1	0	1	0	0	0	0	1	3
U_7	1	0	0	0	0	1	0	0	0	1	0	0	3
U_8	0	0	0	0	0	0	1	1	0	0	0	0	2
U_9	0	0	0	0	0	0	1	1	0	0	0	0	2
U_{10}	1	0	0	0	0	1	0	0	0	1	0	0	3
U_{11}	1	0	0	0	0	1	0	0	0	1	0	0	3
U_{12}	0	0	1	0	0	1	0	0	0	0	0	1	3

Table 3.1

We find that the total unsatisfied demand (T.U.S. demand) corresponding to thesolution Y^4 is minimum. Therefore,

$X^* = (40, 1, 40, 1, 42, 0, 40, 0, 42, 0, 41, 0)$

Step 3. The new demand vector for the CSS problem is

$b = (0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0)$.

Solve the following LP, with this demand vector,

$$\text{Min.} \quad \sum_{j=1}^{12} x_j$$

s.t.

$$x_2 + x_3 + x_5 + x_6 + x_7 + x_{12} \geq 1$$

$$x_1 + x_3 + x_4 + x_6 + x_7 + x_8 \geq 1$$

$$x_j \geq 0.$$

After solving this LP, we get,

$X = (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

Since this is an integer solution, if we round it off in 12 ways, we get the same solution. The total unsatisfied demand is zero with this solution. Thus $Y^p = X$. Therefore,

$$X^* = X^* + Y^p$$

$$= (40, 1, 41, 1, 42, 0, 40, 0, 42, 0, 41, 0)$$

This is the final solution to the given CSS problem. In this case, this is also an optimum solution.

CHAPTER III

EXACT ALGORITHMS

3.1 Introduction.

In this chapter, we develop two exact algorithms for solving the CSS problem. In Section 3.3, we develop a branch and bound algorithm for solving the CSS problem optimally. This algorithm is an adaptation of the general integer programming branch and bound algorithm. Some of the important features of the proposed algorithm are: (1) Sensitivity analysis is made to impose lower or upper bound type of constraint on any variable to significantly reduce the storage and computational time requirements, (2) The LP-based heuristic algorithm developed in Chapter II is applied at every candidate problem to get a good feasible solution. We first explain in detail, different strategies that are followed in the algorithm. The flow-chart for the proposed branch and bound algorithm is presented. A step-by-step procedure, to perform sensitivity analysis for imposing lower (upper) bound constraint on some variable, is then given. A numerical example is also solved. In Section 3.4, we develop a dual fractional cutting plane algorithm for solving the CSS problem. We introduce two types of cuts in this algorithm. One cut is based on the objective function value and the other one is Gomory's fractional cut.

3.2 Notations:

To simplify discussion in this chapter, we make use of the following additional symbols and notations.

Z	optimum objective function value of the continuous valued CSS problem.
Z_*	lower bound of the solution for the CSS problem.
UB	upper bound of the solution for the CSS problem.
$\bar{Z}$	incumbent objective function value.
X	incumbent solution.
X^k	continuous valued solution of candidate problem P^k.
Z^k	optimum objective function value of the continuous valued candidate problem P^k.
Z_I^k	objective function value of the integer feasible solution obtained when the LP-based heuristic algorithm is applied by taking X^k as the first LP solution.
X_I^k	Integer feasible solution obtained when the LP-based heuristic is applied at candidate problem P^k.
L	array containing lower bounds on variables x_1, x_1, $i = 1, \ldots, m+n$.
U	array containing upper bounds on variables x_1, $i = 1, \ldots, m+n$.
B	{b1, b2, ..., bm} set of all basic variables, where bi denotes the index of the basic variable corresponding to the i-th row.

3.3 Branch and Bound Algorithm:

The basic idea in this method is to investigate all the feasible solutions by partitioning the solution set into subsets by a process, known as branching. A lower bound is calculated for the solutions with in each subset. If the lower bound is smaller than the objective function value of the incumbent solution and the subset with the lower bound is feasible, further branching is stopped from this subset. This is known as fathoming. After each partitioning, those subsets with a bound that exceeds that of a known feasible solution are excluded from further partitionings. This is known as pruning. The partitioning continues until a feasible solution is found such that its value is not greater than the bound for any subset. The algorithm terminates when all the subsets are either pruned or fathomed. We now describe various strategies of the branch and bound algorithm for the CSS problem considered in this thesis.

A branch and bound algorithm is characterized by its (i) bounding strategy, (ii) fathoming strategy, (iii) pruning strategy, (iv) branching strategy, and (v) search strategy. These strategies for the suggested branch and bound algorithm for the CSS problem are discussed in the following sub-sections.

3.3.1 Bounding Strategy:

First, we solve the given CSS problem by relaxing the integrality restriction on all variables. The lower bound for

the given CSS problem is the smallest integer value greater than or equal to the continuous valued objective function.

$$LB = \underline{Z} = \lceil Z \rceil .$$

The lower bound for candidate problem $P^k = \lceil Z^k \rceil$.

Initially we do not have any incumbent, therefore,

$$\bar{Z} = \infty .$$

3.3.2 Fathoming Strategy:

We fathom a candidate problem P^k, if Z_I^k satisfies the following condition.

$$\lceil Z^k \rceil = Z_I^k$$

3.3.3 Pruning Strategy:

We prune a candidate problem P^k, if either of the following conditions are satisfied

$$\lceil Z^k \rceil \geq \bar{Z} , \text{ or}$$

candidate problem P^k is infeasible.

3.3.4 Branching Strategy:

If a candidate problem P^k is neither fathomed nor pruned, it is used for subsequent branching. To perform branching at a candidate problem P^k, we can follow different strategies. The variable selected for branching should be such that it results in the enumeration of minimum number of nodes for finding the optimum solution. Many strategies were computationally tried by us.

The following strategy produced consistently better results than the other strategies.

As observed earlier in Section 2.2, by rounding-off the solution obtained at candidate problem P^k, using (2.2.1.1) to (2.2.1.n) we get n integer solutions. Let Y^p be the solution with minimum amount of total unsatisfied demand and U^p be the unsatisfied demand vector corresponding to this solution. If $\sum_{i=1}^{m} u_i^p$ is zero we fathom this candidate problem otherwise we select a variable x_s for further branching. The strategy used for selecting x_s is explained below.

We have U^p available with us. Now, find out, F_j, the number of days on which shift j is active during the days for which demand is not satisfied. For each of the basic variables we compute, $f_j = \lceil x_j \rceil - x_j$. We select a variable x_s such that $F_s \times f_s \geq F_j \times f_j$, $\forall\ j = 1, \ldots, n$. It is observed that selecting a variable such as x_s for branching brings in maximum amount of change in the lower bound in most of the cases. Now, we give a step by step procedure to identify branching variable x_s.

Step 0: Set $E_j = F_j = f_j = 0, \quad \forall\ j = 1, \ldots, n$

Step 1. For all $i = 1, \ldots, m$ if $(u_i^p > 0)$ then

for all $j = 1$ to n such that $a_{ij} = 1$, set $F_j = F_j + 1$,

for all $j = 1$ to n such that $j \in B$, set $f_j = \lceil x_j \rceil - x_j$.

For all $j = 1$ to n such that $j \in B$, set $E_j = F_j \times f_j$.

Find out a variable x_s such that

$$E_s = \underset{1<j<n}{\text{maximum}} \{E_j\}$$

Select the variable x_s as the source variable for further branching.

It is computationally observed that imposing upper bound type of constraint first results in the enumeration of smaller number of nodes for most of the problems. Hence, this strategy has been implemented.

3.3.5 Search Strategy:

As far as searching strategy is concerned, depth first search is employed because of the ease in implementation and lesser storage requirements.

We give the flow chart of the proposed branch and bound algorithm in Fig. 3.1.

3.3.6 Sensitivity Analysis:

In this section, we present a step by step procedure to perform sensitivity analysis by implicitly considering lower bound or upper bound type of constraint on some variable x_s. Let B denote the set of all basic variables at any stage of the problem. Corresponding to this basis, the relative cost coefficients of the variables x_j, $j = 1, \ldots, (m+n)$ are $\bar{c}_j = c_j - \sum_{i=1}^{n} \bar{a}_{ij}$. Set $G = \{j: \bar{c}_j \geq 0\}$ and $H = \{j: c_j < 0\}$ where G and H respectively denote the variables at their lower bound and at their upper bounds

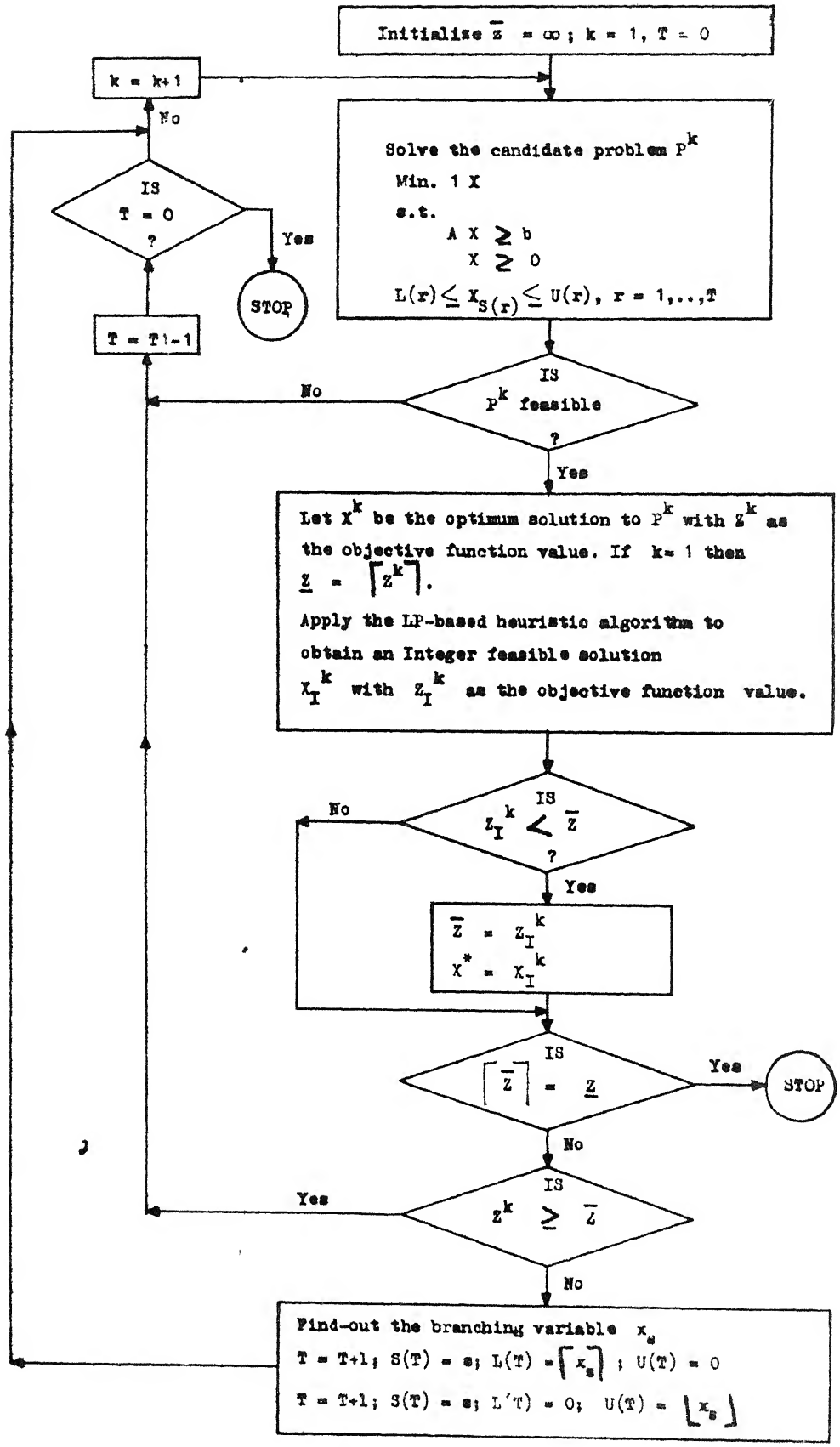

Initialize $\bar{z} = \infty$; k = 1, T = 0
k = k+1
No
IS T = 0 ?
Yes
STOP
Solve the candidate problem P^k
Min. 1 X
s.t.
A X $\geq$ b
X $\geq$ 0
$L(r) \leq X_{S(r)} \leq U(r)$, r = 1,..,T
T = T-1
IS P^k feasible ?
No
Yes
Let X^k be the optimum solution to P^k with Z^k as the objective function value. If k= 1 then $\underline{Z} = \lceil Z^k \rceil$.
Apply the LP-based heuristic algorithm to obtain an Integer feasible solution X_I^k with Z_I^k as the objective function value.
IS $Z_I^k < \bar{Z}$?
No
Yes
$\bar{Z} = Z_I^k$
$X^* = X_I^k$
IS $\lceil \bar{Z} \rceil = \underline{Z}$
Yes
STOP
No
IS $Z^k \geq \bar{Z}$
Yes
No
Find-out the branching variable x_s
T = T+1; S(T) = s; L(T) = $\lceil x_s \rceil$; U(T) = 0
T = T+1; S(T) = s; L(T) = 0; U(T) = $\lfloor x_s \rfloor$

FIG. 3.1

Let (B, G, H) be an optimal basis structure of the CSS problem. Let $A = \{\bar{a}_{ij}\} = B^{-1} A$. Further, let δ be the amount of change required in the current value of the variables x_s to satisfy the bound imposed on it. We first try to make the variable x_s non-basic, if it is a basic variable. If x_s is a basic variable and no variable is found to be eligible to enter the basis, the problem is infeasible. If $s \in B$ and there is some variable t eligible to enter the basis, we make the variable x_s non-basic by performing one dual simplex iteration [20]. Now, we find out the maximum amount of required change that can be brought in the value of the variable x_s without changing the present basis structure. We denote this possible change by λ. If λ is greater than the required change δ, we bring-in δ amount of change in the value of the variable x_s and update the b column for the corresponding change and stop. If λ is less than δ, only λ amount of change can be made in the value of the variable x_s without changing the present basis. We bring λ amount of change in the value of x_s and update $\bar{b}$ column for the corresponding change brought in x_s. In this case, the bound restriction on the variable x_s is not yet satisfied because one of the basic variables reaches either its lower bound or its upper bound before the required change in the value of x_s can be brought in. The dual simplex iteration is again performed to replace the basic variable, which has reached its lower or upper bound recently, by an appropriate non-basic variable. The

basis structure is updated and this process is repeated until either $\delta = 0$ or it is found that the dual simplex iteration can not be performed indicating infeasibility of the problem. A formal description of the procedure to perform sensitivity analysis is given below.

We consider the case when some lower bound restriction is imposed on variable x_s. We give the necessary changes required to be done if upper bound type of restriction is to be imposed on x_s.

Step 0: Let B denote the set of all basic variables, $G = \{j: \bar{c}_j \geq 0\}$, and $H = \{j: \bar{c}_j < 0\}$, where, $\bar{c}_j = c_j - \sum_{i=1}^{m} a_{ij}$, $\forall\ j = 1, \ldots, n$. Let L and U be two arrays respectively containing the tightest lower bound and upper bound restrictions on each variable j, $\forall\ j = 1, \ldots, m+n$.

Set $L(s) = x_s$.

If $s \notin B$ go to Step 2.

Let r be the row corresponding to x_s in $\bar{A}$.

Step 1: The variable x_{br} leaves the basis at its lower bound or upper bound. If x_{br} is equal to L(br) go to 1(a), else go to 1(b).

1(a): Let $K = \{j: j \in G \text{ and } \bar{a}_{rj} < 0, \text{ or } j \in H \text{ and } a_{rj} > 0\}$. If K is empty, go to Step 3, otherwise identify a non-basic variable x_t for which

$-\bar{c}_t/\bar{a}_{rt} = \min_{j\in K} \{-\bar{c}_j/\bar{a}_{rj}\}$, and go to 1(c).

1(b): Let $K = \{j: j \in G \text{ and } \bar{a}_{rj} > 0, \text{ or } j \in H \text{ and } \bar{a}_{rj} < 0\}$. If K is empty, go to Step 3, otherwise identify a non-basic variable x_t for which

$\bar{c}_t/\bar{a}_{rt} = \min_{j\in K} \{\bar{c}_j/\bar{a}_{rj}\}$, and go to 1(c).

1(c): The variable x_{br} leaves the basis and x_t enters the basis. Update the $\bar{A}$ matrix (except the column corresponding to $\bar{b}$) by performing one dual simplex iteration. Go to Step 2.

Step 2: Update all $x_j \in B$ as follows:

Compute $\delta = \lceil L(s) \rceil - L(s)$

If δ is zero go to Step 4, otherwise compute,

$$\lambda_1 = \begin{cases} (\bar{b}_1 - L(b_1))/\bar{a}_{is}, & \text{if } \bar{a}_{is} > 0 \\ \infty, & \text{if } \bar{a}_{is} = 0 \\ -(U(b_1) - \bar{b}_1)/\bar{a}_{is}, & \text{if } \bar{a}_{is} < 0 \end{cases}$$

Let $\lambda = \min_{1\le i\le m} \{\lambda_i, \delta\}$

Now update $x_{bi} = x_{bi} - \bar{a}_{is} \cdot \lambda$

$x_s = x_s + \lambda$

$L(s) = L(s) + \lambda$

If $\delta = \lambda$ go to Step 4 otherwise identify the row r corresponding to the minimum λ_i and go to Step 1.

Step 3: The problem is infeasible. STOP.

Step 4: The solution X is optimum and STOP.

For performing sensitivity analysis to impose upper bound type of constraint on variable x_s, we make the following changes in Step 0 and Step 2.

Step 0: Set $U(s) = x_s$ instead of $L(s) = x_s$.

Step 2: Update all $x_j \in B$ as follows.

Compute $\delta = U(s) - \lfloor U(s) \rfloor$

If δ is zero go to Step 4, otherwise,

compute,

$$\lambda_i = \begin{cases} (u(b_i) - \bar{b}_i)/\bar{a}_{is}, & \text{if } \bar{a}_{is} > 0 \\ \infty, & \text{if } \bar{a}_{is} = 0 \\ -(\bar{b}_i - L(b_i))/\bar{a}_{is}, & \text{if } \bar{a}_{is} < 0 \end{cases}$$

Set $\lambda = \min_{1 \le i \le m} \{\lambda_i, \delta\}$

Now update $x_{bi} = x_{bi} + \bar{a}_{is} \cdot \lambda$

$x_s = x_s - \lambda$

$U(s) = U(s) - \lambda$

If $\bar{\delta} = \lambda$ go to Step 4, otherwise identify the row r corresponding to the minimum λ_i and go to Step 1.

If a node in the branching tree is either fathomed or pruned we have to get back to its parent, for this standard bounded linear programming techniques [15] can be used.

The algorithm is coded in FORTRAN-IV on DEC-1090 computer system and the listings are enclosed at the end. Extensive computational investigation is done. The performance of the algorithm is quite encouraging and is discussed in Chapter IV. We now take up a numerical example to illustrate the method.

3.3.7 Numerical Example:

We take up the same example that is considered in Chapter II. The branching tree enumerated is given in Fig. 3.2. Nodes are numbered by the order in which they are enumerated. The branching variable and the bound imposed on it is shown at each node. The lower bound obtained at each node is written in a rectangle at that node. Initially, the given CSS problem is solved as an ordinary LP, the continuous valued objective function value obtained is $Z^1 = 247.00$ $\lceil Z^1 \rceil = \lceil 247.00 \rceil = 247$ is the lower bound for the CSS problem. When LP-based heuristic is applied at node 1, we get a feasible solution with $Z_I^1 = 248$. Therefore, $\bar{Z} = 248$. The algorithm terminated after enumerating 25 nodes. In this problem the initial solution obtained at node 1 itself is the optimum solution.

$$X = (40, 1, 41, 1, 42, 0, 40, 0, 42, 0, 41, 0).$$

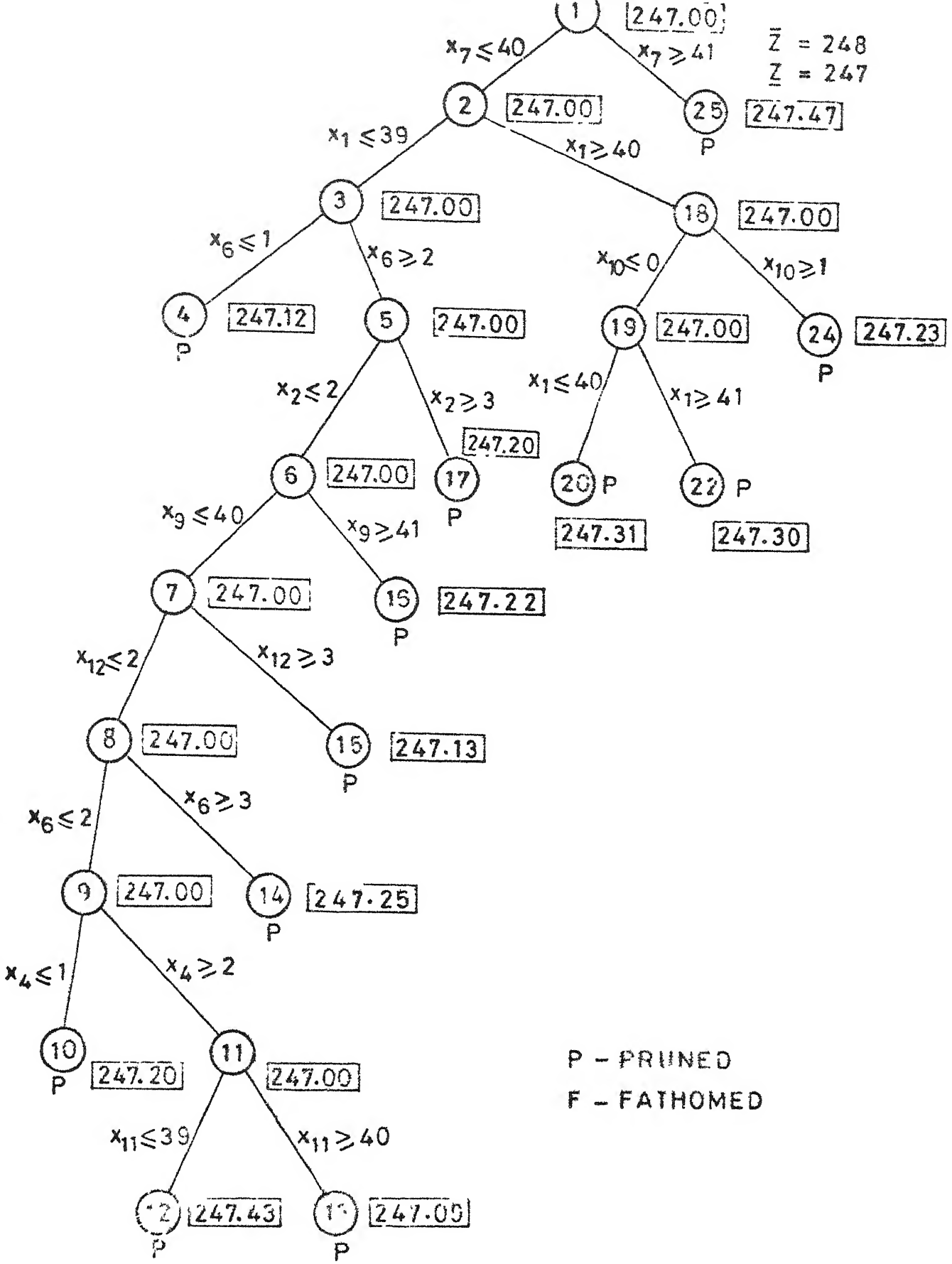

1
247.00
$\bar{Z}$ = 248
$\underline{Z}$ = 247
$x_7 \leq 40$
$x_7 \geq 41$
2
247.00
25
247.47
P
$x_1 \leq 39$
$x_1 \geq 40$
3
247.00
18
247.00
$x_6 \leq 1$
$x_6 \geq 2$
$x_{10} \leq 0$
$x_{10} \geq 1$
4
247.12
P
5
247.00
19
247.00
24
247.23
P
$x_2 \leq 2$
$x_2 \geq 3$
$x_1 \leq 40$
$x_1 \geq 41$
247.20
6
247.00
17
P
20
P
22
P
247.31
247.30
$x_9 \leq 40$
$x_9 \geq 41$
7
247.00
16
247.22
P
$x_{12} \leq 2$
$x_{12} \geq 3$
8
247.00
15
247.13
P
$x_6 \leq 2$
$x_6 \geq 3$
9
247.00
14
247.25
P
$x_4 \leq 1$
$x_4 \geq 2$
10
P
247.20
11
247.00
$x_{11} \leq 39$
$x_{11} \geq 40$
12
247.43
P
13
247.09
P
P - PRUNED
F - FATHOMED

FIG. 3.2

3.4 Cutting Plane Algorithms

3.4.1 Introduction:

In this section, we develop a dual fractional cutting plane algorithm for solving the CSS problem. The general intent of cutting plane algorithms is to deduce supplementary **inequalities** from the integrality and constraint requirements and to append to the constraint set so as to eventually produce a linear program whose optimal solution is integer in the integer constrained variables. Gomory [17] developed the first cutting plane algorithm applicable to any Integer Program. Gomory [17] produced a second cutting plane algorithm for the integer program which requires only additions and subtractions in computation (an "all-integer" technique). All these methods maintain linear programs which are dual feasible, and are therefore often classified as dual cutting plane algorithms.

Glover and Young developed cutting plane algorithms for the Integer Program which maintain linear programs that are primal feasible. Since primal feasible integer solutions are successively produced, the technique is referred to as a primal cutting plane algorithm.

In this dissertation, we consider a cutting plane algorithm for the CSS problem which utilizes the lexicographic dual simplex method and allows fractional numbers in computation - hence the dual fractional reference. We take up the dual

fractional cutting plane algorithm because of its faster convergence rate.

3.4.2 Development of the Algorithm:

To avoid cycling and to ensure convergence we make use of lexicographic dual simplex method in the cutting plane algorithm.

Consider the problem,

$$\text{Max.} \quad C X = Z$$

s.t.

$$A X \leq b, \quad X \geq 0, \text{ integer}$$

This problem is represented in the following way in an inverse tableau to make use of lexicographic dual simplex method.

	1	$-x_1$	$\cdots$	$-x_j$	$\cdots$	$-x_n$	
$x_o =$	a_{oo}	a_{o1}	$\cdots$	a_{oj}	$\cdots$	a_{on}	
$x_1 =$	0	-1	$\cdots$	0	$\cdots$	0	≥ 0
$\vdots$							$\vdots$
$x_j =$	0	0	$\cdots$	-1	$\cdots$	0	≥ 0
$\vdots$							$\vdots$
$x_n =$	0	0	$\cdots$	0	$\cdots$	-1	≥ 0
$x_{n+1} =$	$a_{n+1,o}$	$a_{n+1,1}$	$\cdots$	$a_{n+1,j}$	$\cdots$	$a_{n+1,n}$	≥ 0
$\vdots$							$\vdots$
$x_{n+m} =$	$a_{n+m,o}$	$a_{n+m,1}$	$\cdots$	$a_{n+m,j}$	$\cdots$	$a_{n+m,n}$	≥ 0

where

$$x_o = Z,$$

$$a_{oo} = o, \quad a_{oJ} = -c_J, \qquad \forall\ J = 1,\ldots, n$$

$$a_{n+i,o} = b_i, \qquad \forall\ i = 1,\ldots, m$$

$$a_{n+i,J} = a_{iJ}, \qquad \forall\ i = 1,\ldots, m,\ \forall\ J = 1,\ldots, n$$

In addition, we use the following notation:

J contains the n indices of the current non-basic variables as they appear from left to right on the top most line of the tableau. Initially, $J = \{1,2,\ldots, n\}$.

$J(j)$ $(j = 1,\ldots,n)$ is the j-th element in J.

Now, we explain how our cutting plane algorithm proceeds. We first solve the given CSS problem as an LP using lexicographic dual simplex method [15] and then round-off the continuous valued solution obtained in n ways using (2.2.1.1) to (2.2.1.n) to obtain $Y^1,\ldots, Y^n$. Let Y^p be the solution satisfying the following inequality:

$$\sum_{i=1}^{m} u_i^p \leq \sum_{i=1}^{m} u_i^J, \quad \forall\ J = 1,\ldots, n.$$

If $U^p = 0$, then Y^p is an optimum solution to the CSS problem and we stop. Otherwise, we impose a cut which eliminates the current non-integer solution and is satisfied by all the integer feasible solutions to the given CSS problem. We propose to impose two types of cuts. One cut is based on the

objective function value ($\bar{Z}$) of the optimum solution and the other cut is Gomory's fractional cut.

Since the CSS problem is a pure integer program, the objective function value of the optimum solution to this problem must be an integer. Therefore, whenever we observe $\lceil \bar{Z} \rceil > \bar{Z}$, we append the following cut to inverse tableau.

$$\sum_{k=1}^{n} x_k \geq \lceil \bar{Z} \rceil \qquad (3.4.2.1)$$

However, this cut can not be added directly in the above form. This problem can be tackled in the following way. Initially, before solving the LP, we append one more additional constraint $\sum_{j=1}^{n} x_j \geq 0$ in the $(m+n+1)$-th row and then solve the LP. Let x_{m+n+1} be the slack variable corresponding to this constraint. We show that (3.4.2.1) is equivalent to,

$$x_{m+n+1} = \bar{Z} - \lceil \bar{Z} \rceil + \sum_{j=1}^{n} a_{m+n+1,j} \, (-x_{J(j)}) \qquad (3.4.2.2)$$

The row associated with x_v in the current inverse tableau leads to constraint,

$$x_v = \bar{b}_v + \sum_{j=1}^{n} \bar{a}_{vj} \, (-x_{J(j)}) \qquad (3.4.2.3)$$

Therefore, from (3.4.2.1) and (3.4.2.3)

$$\sum_{k=1}^{n} \bar{b}_k + \sum_{j=1}^{n} \left(\sum_{j=1}^{n} \bar{a}_{kj} \right) (-x_{J(j)}) \geq \lceil \bar{Z} \rceil$$

or,

$$\sum_{k=1}^{n} \bar{b}_k + \sum_{j=1}^{n} \left(\sum_{k=1}^{n} \bar{a}_{kj} \right) (-x_{J(j)}) \geq \lceil \bar{Z} \rceil$$

therefore,

$$\bar{Z} + \sum_{j=1}^{n} a_{m+n+1,J} \, (-x_{J(J)}) - x_{m+n+1} = \lceil \bar{Z} \rceil$$

$$x_{m+n+1} = \bar{Z} - \lceil \bar{Z} \rceil + \sum_{J=1}^{n} a_{m+n+1,j} \, (-x_{J(J)}).$$

Therefore, whenever we have to impose a constraint like (3.4.2.1), we simply make $\bar{a}_{m+n+1,o} = (\bar{Z} - \lceil \bar{Z} \rceil)$. Now (m+n+1)-th constraint violates primal feasibility because $(\bar{Z} - \lceil \bar{Z} \rceil)$ is always negative. Apply dual simplex method to make the tableau primal feasible.

The second type of cut we impose is the standard Gomory's fractional cut. We impose this cut when $\bar{Z}$ is an integer, but none of the n rounded-off solutions are feasible to the given CSS problem. If x_v is the source variable the following constraint is appended at the bottom of the inverse tableau.

$$x_{(m+n+1)+k} = -f_{vo} + \sum_{J=1}^{n} (-f_{vJ}) \, (-x_{J(J)}) \geq 0 \qquad (3.4.2.4)$$

where, $x_{(m+n+1)+k}$ is Gomory's slack variable associated with k-th fractional cut added.

$$F_{vJ} = a_{vJ} - \lfloor a_{vJ} \rfloor, \qquad \forall \; J = 0,1,\ldots, n$$

3.4.3 Source Variable Selection:

Proper selection of source variable for cut generation plays very important role in getting a fast optimal solution.

Let x_v be the source row. The hyperplane defining the cut is,

$$\sum_{j=1}^{n} f_{ij}\, x_{J(j)} = f_o.$$

We may say the larger the value of the ratio f_o/f_{ij} the stronger the cut is [17]. Hence, the objective when selecting a source row is to produce an inequality or hyperplane in which these ratios are as large as possible. Since source rows will usually yield inequalities in which the f_o/f_{ij} ratios are large for some variables and small for others, it is not clear as to which row to select. However, we intuitively feel that the following strategy produces a deep cut.

Step 1: Set $f_i = \bar{b}_i - \lfloor b_i \rfloor$, $\forall$ $i = 1, \ldots, m+n$.
and $F_1 = 0$.
Arrange the f_1's in decreasing order of their magnitude and select the first t f_1's.

Step 2. For each of the t f_1's selected in Step 1
Compute $\sum_{J=1}^{n} f_{1J}$ and N_i, the total number of non-zero f_{1J}'s in the 1-th row. Set $F_1 = \sum \frac{f_i}{f_{iJ}} \quad N_1$

Let $F_v = \max_{1 \le i \le m+n} \{F_i\}$

Select the variable x_v as the source variable for cut generation.

We have computationally observed that if the value of t is between 10 and 20, it would give better results. A formal statement of the algorithm is presented in the next section.

3.4.4 Statement of the Algorithm:

Step 1: Solve the following LP using Lexico-dual simplex method after transforming the given CSS problem into the inverse tableau form.

$$\text{Min.} \quad 1\ X$$

s.t.

$$A\mathbf{X} \geq b$$

$$\sum_{J=1}^{n} X_J \geq 0$$

$$X \geq 0$$

Let $\bar{X}$ be the optimum solution obtained.

Step 2: Use n round-offs (2.2.1.1) to (2.2.1.n) to get $Y^1, \ldots, Y^n$. Select Y^p satisfying the following inequality,

$$\sum_{i=1}^{m} u_i^p \leq \sum_{i=1}^{m} u_i^J, \qquad \forall\ J = 1, \ldots, n$$

If $U^p = 0$, go to Step 5, otherwise go to Step 3.

Step 3: If $\lceil \bar{Z} \rceil = \bar{Z}$ go to Step 4. Otherwise, introduce the cut (3.4.2.2). Perform necessary dual simplex iterations to get primal feasibility and go to Step 2.

Step 4: Find out the source row variable x_v using the routine described above and append the cut (3.4.2.3) to the bottom of the inverse tableau. Perform the dual simplex

iterations to make the problem primal feasible.

Go to Step 2.

Step 5: Y^p is the optimum solution to the CSS problem. STOP.

The computational performance of the algorithm is discussed in Chapter IV.

CHAPTER IV

COMPUTATIONAL INVESTIGATIONS

4.1 Introduction:

In this chapter, we present the computational performance of LP-based heuristic algorithm developed in Chapter II, the branch and bound algorithm and the dual fractional cutting plane algorithm developed in Chapter III and Bartholdi's guaranteed accuracy round-off algorithm.

The computational experiments require a large number of problems. The details of the problems considered for computational study are given in Section 4.2. In Section 4.3, we discuss the various considerations such as accuracy, CPU time for analysing the three algorithms developed by us and Bartholdi's algorithm. Finally, in Section 4.4, the concluding remarks are given.

4.2 Problems Considered for Experimentation:

We have considered five different sizes of problems for the computational study. The five sizes are (12 x 12), (24 x 24), (48 x 48), (72 x 72), and (96 x 96). To explain the significance of the problem size, for example, let us consider the problem of (24 x 24) size. These numbers indicate

that the planning horizon for scheduling is 24 time periods and there are 24 different types of shifts available for assigning the workers. The motivation behind considering these particular problem sizes is as follows. Normally, most of the service organizations such as telephone exchanges, hospitals, police and fire departments work round the clock. Depending on the demand, they can have a different shift starting every two hours, one hour, half an hour or quarter hour in a day. Thus, these particular sizes are considered.

As we have already discussed in Chapter I, a shift is differentiated from the other shifts by its start time or by its length or by the number of breaks or by the timing of breaks or by the combination of some or all of these. For a given problem, we differentiate two shifts by their start time in the problems considered. The cycle length, number of blocks of consecutive ones, q, and the timing of breaks are similar for all the n shifts.

Within each size of the problem, we considered the shifts with different values of q. The values of q considered are 2, 4, 6, and 8. For a given q and cycle length, we again considered two cases. In the first case, the cycle is symmetric about its middle elements and in the other case the cycle is non-symmetric. Altogether, 31 problems, 16 symmetric and 15 non-symmetric problems are considered. Tables 4.1 and 4.2 respectively provide the details of the symmetric and non-symmetric problems considered.

A cycle is an (m x 1) column vector. Given a cycle, we can generate the A matrix by taking the (m x 1) column vector and shifting it cyclically one position downward to obtain successive columns. Thus, the A matrix has the form

$$\begin{matrix} a_1 & a_m & \cdots & a_2 \\ a_2 & a_1 & \cdots & a_3 \\ \vdots & \vdots & & \vdots \\ a_m & a_{m-1} & \cdots & a_1 \end{matrix}$$

and is generated by the column vector $(a_1, \ldots, a_m)$.

It is possible that not all the n shifts may be present in practice. As the number shifts becomes smaller, the problem becomes easier to solve. For all the problems are considered, all the n shifts are assumed to be present.

The demands during different time intervals were generated randomly by taking the range as 100 to 150.

In reality, there may be some pattern in the demand. To see how the performance of our algorithms is effected when there is certain pattern in demand, we have considered the following five demand patterns and solved two problems of (48 x 48) size and (72 x 72) size with q = 8. The demand patterns considered are as shown in Fig. 4.1.

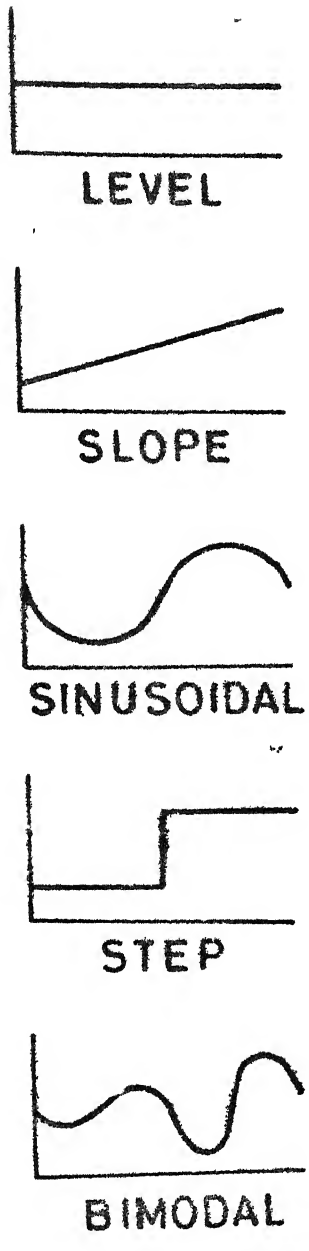


Fig. 4.1 Various types of demand patterns.

Level: Here the demand is constant during all the time periods.

Slope. Here, the demand increases linearly from start time period to the end time period.

Step: In this case, the demand remains constant during first half of the time periods and then shoots up suddenly and remains at that value during rest of the time-periods.

Sinusoidal: In this case the demand pattern is like a sine wave.

Bimodal: Here, the demand pattern is like a sine wave during each half of the total time periods.

In Tables 4.1 and 4.2, we have given certain number for each problem. In the subsequent tables and discussion, these problems are referred to by their numbers.

We can observe that the CSS problem is completely specified, if we are given the problem size, cycle and the range of demands.

Computational Performance:

We, now give the computational performance of our algorithms and Bartholdi's algorithm in terms of accuracy and CPU times.

In Table 4.3, we give the objective function value obtained and the CPU times taken by Bartholdi's method, LP-based heuristic algorithm and the branch and bound algorithm for the symmetric problems. The total number of nodes enumerated in branch and bound algorithm is also given in the table.

It is evident from the table that the branch and bound algorithm was able to obtain optimum solution to all the problems. The maximum number of nodes enumerated are 53 for the problem 14. This problem has also taken the maximum amount of CPU time of 274.4 seconds.

The objective function values obtained using LP-based heuristic are observed to be very close to the optimum value in all the problems. The deviation from the optimum solution is less than 0.5 percent. The CPU time taken is much lesser when compared to branch and bound algorithm. This algorithm is observed to perform far better than Bartholdi's method in terms of accuracy especially when q is more.

It was observed that for none of the problems, the LP-based heuristic solved more than 2 LP's. In addition, the size of the second LP solved was often less than 10 percent of the original problem size.

Bartholdi's method was observed to be often taking as many additional units as given by his bound. It is better only in terms of the CPU time taken.

Similar data for the non-symmetric problems is given in Table 4.4.

The performance of branch and bound is somewhat discouraging in this case. 6 of the 15 problems were solved optimally by

enumerating less than 25 nodes. In those cases where the algorithm did not terminate after enumerating 100 nodes are marked with a star. For these problems we used ε - approximation with $\varepsilon = 1$. In this case, we terminate the problem if the incumbent solutions $[\bar{Z}]$ satisfies the following condition:

$\bar{Z}$ is less than or equal to $(\underline{Z} + \varepsilon)$.

The LP-based heuristic and Bartholdi's method behaved in the same way as they did for the symmetric problems.

Problems 8 and 12 were solved for each of the five demand patterns described in the earlier section and the results are given in Table 4.6. In these cases, the LP-based heuristic algorithm performance is same as for the earlier problems. The branch and bound algorithm was able to solve problem 8 optimally for all the demand patterns. Problem 12 could not be solved when the demand pattern is constant or sinusoidal. When 1 unit of ε - approximation is applied, these two cases were also solved. From the nodes enumerated, we can observe that the problem becomes more difficult to solve when there is certain pattern in demand.

The performance of the dual fractional cutting plane algorithm was quite erratic. The problems upto the size (72 x 72) were tried on this algorithm. Out of the 23 problems tried only 7 problems were solved and the others did not terminate after appending 150 cuts. The results of those problems

which were solved are given in Table 4.7. The algorithm could not solve problem sizes more than (48 x 48). In those cases, it could solve the problem, the number of cuts generated are very less in number.

4.4 Concluding Remarks:

In Chapter I, we observed that the CSS problem with intermittently available resources is more difficult to solve. We gave the statement of the problem and reviewed the literature available for the CSS problem. We have developed three algorithms to solve the CSS problem more accurately.

The LP-based heuristic algorithm developed in Chapter II makes use of an improved version of the rounding-off procedure suggested by Bartholdi et al [5] and solves a sequence of LP's to get a good feasible solution to the CSS problem.

We developed a branch and bound algorithm in Chapter III. The salient features of this algorithm are: (1) It makes use of the LP-based heuristic algorithm at every node to get a good feasible solution, (2) Sensitivity analysis was used to significantly reduce the storage and computational time requirements, and (3) the criterion used for source row selection makes pruning more effective. We have also developed a dual fractional cutting plane algorithm in Chapter III. We proposed two types of cuts, one cut is based on objective function value and the other cut is Gomory's fractional cut.

Finally, we have done extensive computational study on all these three algorithms and Bartholdi's algorithm. We observed that the performance of the LP-based heuristic algorithm was quite encouraging. The deviation of the solution obtained using this algorithm from the optimum solution was not more than 2 units generally. We observed that not more than 2 LP's were needed to be solved for any problem in the LP-heuristic algorithm and the size of the second LP solved is often less than 10 percent of the original problem size. The performance of the heuristic was invariant with the pattern of demand.

The performance of the branch and bound algorithm is quite encouraging. Symmetric problems of the size (96 x 96) with $q = 8$ were solved optimally in a few minutes of CPU time. We have observed that problem becomes more difficult to solve when the cycle is non-symmetric or there is certain pattern in demand.

The performance of cutting plane algorithm was observed to be erratic.

We have considered the CSS problem when all c_j's are 1. Further scope for research in this area would be to extend the ideas developed in this thesis to consider the case when c_j's are different.

Table 4.1: Symmetric Problems.

Problem Number	Size MxN	q	Cycle Length	k	Speed	Cycle
1	12 x 12	4	7	4	10	1 0 1 0 1 0 1
2	24 x 24	2	9	8	10	1 1 1 1 0 1 1 1 1
3	24 x 24	4	11	8	30	1 1 0 1 1 0 1 1 0 1 1
4	24 x 24	6	13	8	50	1 1 0 1 0 1 0 1 0 1 0 1 1
5	48 x 48	2	18	16	10	1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1
6	48 x 48	4	20	16	30	1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1
7	48 x 48	6	22	16	55	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 1 1 1 0 1 1 1
8	48 x 48	8	24	16	70	1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1
9	72 x 72	2	27	24	10	1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1
10	72 x 72	4	29	24	30	1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
11	72 x 72	6	31	24	50	1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1
12	72 x 72	8	33	24	70	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
13	96 x 96	2	34	30	10	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
14	96 x 96	4	36	30	33	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
15	96 x 96	6	38	30	50	1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1
16	96 x 96	8	40	30	77	1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1

Table 4.2: Non-Symmetric Problems.

Prob. No.	Size MxN	q	Cycle Length	K	Seed	Cycle
17.	24 x 24	2	9	8	20	1 1 1 1 1 0 1 1 1
18.	24 x 24	4	11	8	40	1 1 1 0 1 0 1 1 0 1 1
19.	24 x 24	6	13	8	60	1 0 1 1 0 1 0 1 1 0 1 0 1
20.	48 x 48	2	18	16	20	1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1
21.	48 x 48	4	20	16	40	1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 0 1 1 1 1
22.	48 x 48	6	22	16	60	1 1 1 1 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1
23.	48 x 48	8	24	16	80	1 1 1 0 1 1 0 1 1 0 1 0 0 1 1 1 0 1 1 0 1 1 0 1
24.	72 x 72	2	27	24	20	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1
25.	72 x 72	4	29	24	40	1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 1 1 1 1 1
26.	72 x 72	6	31	24	60	1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1
27.	72 x 72	8	33	24	80	1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 0 1 1 1 0 1 1 1 0 1
28.	96 x 96	2	34	30	20	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
29.	96 x 96	4	36	30	40	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1
30.	96 x 96	6	38	30	60	1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 0 0 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1
31.	96 x 96	8	40	30	80	1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1

Table 4.3: Computational results for symmetric problems.

Prob. No.	Size MxN	Objective Function Value			CPU Time in Secs.			No. of nodes Examined in B and B.
		Bartholdi's Method	Heuristic Method	B and B Algorithm	Bartholdi's Method	Heuristic Method	B and B Algorithms	
1.	12 x 12	408	399	399	< 0.1	< 0.1	0.1	1
2.	24 x 24	404	402	401	0.5	0.5	0.9	5
3.	24 x 24	409	400	400	0.2	0.3	0.3	1
4.	24 x 24	393	380	379	0.4	0.5	4.9	37
5.	48 x 48	395	393	391	3.2	3.7	14.5	18
6.	48 x 48	402	395	393	5.1	5.5	17.4	22
7.	48 x 48	394	380	379	3.6	3.9	22.1	26
8.	48 x 48	411	392	391	4.1	4.3	6.8	5
9.	72 x 72	400	398	397	11.8	13.4	15.0	2
10.	72 x 72	407	399	398	15.4	19.5	64.4	23
11.	72 x 72	416	404	402	19.3	19.6	35.3	9
12.	72 x 72	438	418	417	13.3	19.3	112.1	45
13.	96 x 96	438	436	435	35.7	35.9	44.3	4
14.	96 x 96	437	430	428	36.3	47.4	274.4	53
15.	96 x 96	445	431	430	38.7	49.2	81.7	9
16.	96 x 96	452	432	430	37.4	50.8	245.7	42

Table 4.4: Computational results for non-symmetric problems.

Prob. No.	Size MxN	Objective Function Value			CPU Time in Secs.			No. of nodes Examined in B and B.
		Bartholdi's Method	Heuristic Method	B and B Algorithm	Bartholdi's Method	Heuristic Method	B and B Algorithms	
17.	24 x 24	384	381	381	0.4	0.4	0.4	1
18.	24 x 24	392	384	384	0.4	0.4	0.4	1
19.	24 x 24	400	386	386	0.3	0.5	4.5	23
20.	48 x 48	393	391	390	2.9	3.5	15.5	20
21.	48 x 48	398	390	390	3.0	4.5	>117.1	> 100
22.	48 x 48	404	390	390	4.3	4.6	>103.1	> 100
23.	48 x 48	406	388	386	2.8	3.6	>118.5	> 100
24.	72 x 72	406	404	403	19.2	12.6	45.2	19
25.	72 x 72	399	391	391	17.7	19.5	>409.9	> 100
26.	72 x 72	414	402	400	18.4	19.1	>340.2	> 100
27.	72 x 72	420	401	400	21.5	24.6	>269.8	> 100
28.	96 x 96	444	441	441	30.7	28.7	29.3	1
29.	96 x 96	426	418	418	41.9	44.5	>698.8	> 100
30.	96 x 96	436	423	422	50.9	60.8	>804.8	> 100
31.	96 x 96	432	414	412	39.6	40.6	>911.7	> 100

Table 4.5: Performance of ε - approximation method.

Prob. No.	Prob. Size	Objective Function Value		No. of Nodes Examined		CPU Time	
		B and B without ε- approximation	B and B with ε- approximation	B and B without ε- approximation	B and B with ε- approximation	B and B without ε- approximation	B and B with ε- approximation
20	48 x 48	390	391	20	1	15.5	3.7
21	48 x 48	390	390	>100	1	117.1	4.6
22	48 x 48	390*	390	>100	1	103.1	4.8
23	48 x 48	386*	386	>100	70	118.5	76.9
24	72 x 72	403	404	19	1	45.2	13.1
25	72 x 72	391*	391	>100	1	409.9	20.1
26	72 x 72	400*	400	>100	9	340.2	38.5
27	72 x 72	400*	400	>100	10	269.8	41.3
28	96 x 96	441	441	1	1	29.2	29.4
29	96 x 96	418*	418	>100	1	698.8	45.4
30	96 x 96	422*	422	>100	19	804.8	45.4
31	96 x 96	412*	412	>100	48	911.7	303.6

Table 4.6: Performance of heuristic and branch and bound algorithms for various demand patterns.

Prob. No.	Prob. size	Demand Pattern	Objective Function Value		CPU Time (Sec.)		Nodes examined in B and B algorithm	ε - Approximation Method	
			LP-based heuristic	B and B algorithm	LP-based heuristic	B and B algorithm		CPU time (Sec.)	Nodes examined
8	48 x 48	LEVEL	300	300	3.3	3.3	1		
		SINUSOIDAL	377	376	3.5	17.5	17		
		BIMODAL	393	392	3.8	4.1	3		
		SLOPE	390	389	3.8	23.8	27		
		STEP	391	389	4.1	12.5	10		
12	72 x 72	LEVEL	378	376*	14.7	>566.2	>100	206.7	61
		SINUSOIDAL	378	377*	14.7	>366.1	>100	103.3	31
		BIMODAL	426	423	25.4	250.3	75		
		SLOPE	398	395	22.4	166.8	69		
		STEP	386	384	17.3	151.9	54		

Table 4.7: Performance of cutting plane algorithm.

Prob. No.	Size	No. of cuts on objective function	No. of fractional cuts	CPU time in sec.	Optimum objective function value
1.	12 x 12	0	0	< 0.1	399
2.	24 x 24	0	0	0.4	401
3.	24 x 24	0	0	0.3	400
5.	48 x 48	0	4	8.3	391
6.	48 x 48	0	2	6.7	393
17.	24 x 24	1	13	2.6	381
20.	48 x 48	1	3	11.9	390

REFERENCES AND BIBLIOGRAPHY

1. Ahuja, R.K., "Minimax Linear Programming Problem," to appear in Operations Research Letters.

2. Arthur, Jeffrey, L. and Ravindran, A., "A Multiple Objective Nurse Scheduling Model," AIIE Trans. 13, 55-60, (1981).

3. Baker, K.R., "Scheduling a Full-Time Workforce to Meet Cyclic Staffing Requirements," Mgt. Sci. 20, 1561-1568 (1974).

4. Baker, K.R., "Scheduling Full-Time and Part Time Staff to Meet Cyclic Requirements." OR Quart (UK), 25, 65-76 (1974).

5. Bartholdi, J.J., "A Guaranteed Accuracy Round-Off Algorithm for Cyclic Scheduling and Set Covering," O.R. 29, 501-510, (1981).

6. Bartholdi, J.J., J.B. Orlin and H.D. Ratliff, "Cyclic Scheduling via Integers Programs with Circular Ones," O.R. 28, 1074-1085 (1980).

7. Dantzig, G.B., "A Comment on Eddies Traffic Delay at Toll Booths", OR 28, 1074-1085 (1954).

8. Glover, F., and J.R. Mulvey, "Network Related Schedul Models for Problems with Quasi-Adjacency and Block Adjacency Structure," Working Paper HBS 76-3 (1976), Graduate School of Business Administration, Harvard University.

9. Henderson, W.B., "Mathematical Models for Telephone Operator Shift Scheduling," Ph.D. Dissertation, Kranner Graduate School of Industrial Administration, Purdue University (1974).

10. Henderson, W.B., and Berry, W.L., "Heuristic Methods f Telephone Operator Shift Scheduling: An Experimental Analysis", Mgt. Sci. 22, 1372-1380 (1976).

11. Laporte, G., and Mobert, Y., "Rotating Schedules," EJ 4, 24-30 (1980).

12. Luce, B.J., "A Shift Scheduling Algorithm," Paper Presented at the ORSA National Meeting, San Diego (1973).

13. Mabert, V.A., and Watts, C.A., "A Simulation Analysis of Tour-Shift Construction Procedures,"Mgt. Sci. 28, 520-532 (1982).

14. Morris, J.G., and Showalter, M.J., "Simple Approaches to Shift, Days-Off and Tour Scheduling Problem," Mgt. Sci. 29, 942-950 (1983).

15. Murty, K.G., "Linear and Combinatorial Programming," John Wiley and Sons, Inc., New York, 1976.

16. Oudhensden, D.L., and Wen-Jeng, Wu, "Telephone Operator Scheduling with a Fixed Number of Operators," EJOR 11, 55-59 (1982).

17. Salkin, H.M., "Integer Programming," Addison Wesley Publishing Company, Masachusetts, 1975.

18. Segal, M., "The Operator Scheduling Problem: A Network Flow Approach," OR 22, 808-823 (1974).

19. Shepardson, F., and R.E. Marsten, "A Lagrangean Relaxation Algorithm for the Two Duty Period Scheduling Problem,"Unpublished Technical Report No. 532, The University of Arizona, Tucson, Arizona.

20. Willis, R.J., and Wilson, E.J.G., "Scheduling of Telephone Betting Operators - A Case Study," JORS 34, 991-998 (1983).

21. Wagner, H.M., "The Dual Simplex Algorithm for Bounded Variables," NRLQ 5, 257-262 (1958).

```
C
C
C*****************************************************************
C       LP-BASED HEURISTIC ALGORITHM
C*****************************************************************
        INTEGER BVAR(100),IX(100),TIX(100),DEMAND(100),HIGH,CPUT
        INTEGER ENTER,STPOSN(100),CL,AMOUNT,TAMUNT,TIX2(100),CYCLE(100)
        REAL A(0:100,0:200),RX(200),COST(100),GA(100,100)
        INTEGER SDEM(100),USDEM(100),TUSDEM(100)
        COMMON A,M,N
        READ(23,*)NPROB
        DO 120 III=1,NPROB
        WRITE(24,10)III
   10   FORMAT(1X,'PROBLEM -',I2)
        WRITE(24,20)
   20   FORMAT(5X,'MULTIPLE LP APPROACH')
        READ(23,*)M,N,CL
        WRITE(24,30)M,N
   30   FORMAT(5X,'M = ',I3,'  N = ',I3)
        DO 40 J=1,N
        A(0,J)=0
   40   IF(J.GE.N)A(0,J)=1
        READ(23,*)ISEED,LOW,HIGH
        WRITE(24,50)ISEED,LOW,HIGH
   50   FORMAT(5X,'SEED = ',I3,'   RANGE OF DEMANDS IS ',I3,' to ',I3)
        READ(23,*)(CYCLE(I),I=1,CL)
        WRITE(24,55)(CYCLE(I),I=1,CL)
55      FORMAT(5X,'CYCLE:  ',3612)
        CALL GENRAT(ISEED,LOW,HIGH,CYCLE,CL,GA,DEMAND,STPOSN)
        READ(23,*)(DEMAND(I),I=1,M)
        DO 44 I=1,M
44      A(I,NB)=-DEMAND(I)
        A(0,J)=0.
        TC=0.
        MO=M
        DO 60 I=1,M
        DO 60 J=1,N
   60   A(I,J)=-GA(I,J)
        CALL RTIME(N1)
        NA=N+M
        NB=0
        DO 70 I=1,M
   70   TIX(I)=0
        BIGM=10000.
        EPS=0.0001
        NOLPS=0
        ITER=0
   80   NOLPS=NOLPS+1
        DO 100 J=N+1,NA
        DO 90 I=1,M
        A(I,J)=0
   90   IF((J-N).EQ.I)A(I,J)=1
        A(0,J)=0
  100   BVAR(J-N)=J
        A(0,NB)=0
  110   ITER=ITER+1
        BMIN=-EPS
        DO 120 I=1,M
        IF(A(I,NB).GE.BMIN)GO TO 120
        LEAVE=I
        BMIN=A(I,NB)
  120   CONTINUE
        IF(BMIN.EQ.-EPS)GO TO 170
        RMIN=BIGM
        DO 130 J=1,NA
        IF(A(LEAVE,J).GT.-EPS)GO TO 130
        RATIO=-A(0,J)/A(LEAVE,J)
        IF(RATIO.GE.RMIN)GO TO 130
        RMIN=RATIO
        ENTER=J
  130   CONTINUE
        BVAR(LEAVE)=ENTER
        FACTOR=A(LEAVE,ENTER)
        DO 140 J=0,NA
  140   A(LEAVE,J)=A(LEAVE,J)/FACTOR
        DO 160 I=0,M
        IF(I.EQ.LEAVE)GO TO 160
        X=A(I,ENTER)
        DO 150 J=0,NA
  150   A(I,J)=A(I,J)-X*A(LEAVE,J)
  160   CONTINUE
        GO TO 110
  170   CONTINUE
        IF(NOLPS.GT.1)GO TO 190
        XY=-A(0,NB)
C
C
```

```
      30    STPOSN(J)=STP*(J-1)+1
            CALL SETRAN(ISEED)
            DO 40 I=1,M
            U1=RAN(X)
            IDEM=LOW+(HIGH-LOW)*U1+0.5
            DEMAND(I)=IDEM
            A(I,J)=-IDEM
      40    CONTINUE
            RETURN
            END
```

```
C
C*************************************************************************
C
C            BRANCH AND BOUND ALGORITHM FOR
C            CYCLIC STAFF SCHEDULING PROBLEM
C
C*************************************************************************
C        IX-array containing final solution.
C                 -1  FOR VARIABLES AT THEIR LOWER BOUND
C        STATUS =  0 FOR BASIC VARIABLES
C                  1  FOR VARIABLES AT THEIR UPPER BOUND
         INTEGER BVAR(100),DEMAND(100),ENTER,STATUS(200),TOP,SR,UBOBJ,OB
         INTEGER STACK(300,3),IX(100),SV,HIGH,FEAS,BVAR1(100),CYCLE(100)
         INTEGER SA(100,100),IX1(100),TC,STPOSN(100),SDEM1(100),USDEM(100
         IADJUST,CL,TIX(100),USDEM1(100)
         REAL A(101,200),XX(200),COST(100),UB(200),LB(200),A1(101,200)
         COMMON A,BVAR,STATUS,UB,LB,NA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
        1SA,STPOSN,SDEM,USDEM,CL,DEMAND
         READ(21,*)NPROB
         DO 310 IIP=1,NPROB
         WRITE(22,10)IIP
   10    FORMAT(1X,'PROBLEM -',I2)
         READ(21,*)M,N,CL
         READ(21,*)ISEED,LOW,HIGH
         READ(21,*)(CYCLE(I),I=1,CL)
         WRITE(22,15)(CYCLE(I),I=1,CL)
   15    FORMAT(5X,'CYCLE: ',36I2)
         WRITE(22,20)M,N
   20    FORMAT(5X,'M = ',I4,' N = ',I4)
         WRITE(22,30)ISEED,LOW,HIGH
   30    FORMAT(5X,'SEED = ',I4,' DEMAND RANGE IS ',I5,' to',I4)
         MC=M+1
         NA=M+N
         NB=M+N+1
         CALL GENRAT(ISEED,LOW,HIGH,CYCLE)
         ALPHA=0
         DO 44 I=1,M
   44    A(I,NB)=-DEMAND(I)
         CALL RITME(T1)
         DO 40 J=1,NB
         A(MC,J)=0
   40    IF(J.GT.N)A(MC,J)=1
         DO 60 I=1,M
         DO 50 J=1,N
   50    A(I,J)=-SA(I,J)
   60    CONTINUE
         BIG=1000000.
         XLARGE=1000000.
         EPS=.0001
         ITER=0
         DO 70 I=1,NA
         STATUS(I)=-1
         IF(I.GT.N)STATUS(I)=0
         LB(I)=0.
   70    UB(I)=XLARGE
         DO 90 J=N+1,NA
         DO 80 I=1,M
         A(I,J)=0
         IF((J-N).EQ.I)A(I,J)=1
   80    CONTINUE
         A(MC,J)=0
         BVAR(J-N)=J
   90    CONTINUE
         A(MC,NB)=0
  100    ITER=ITER+1
         BMIN=-EPS
         DO 110 I=1,M
         IF(A(I,NB).GE.BMIN)GO TO 110
         LEAVE=I
         BMIN=A(I,NB)
  110    CONTINUE
         IF(BMIN.GE.-EPS)GO TO 160
         RMIN=BIG
         DO 120 J=1,NA
         IF(A(LEAVE,J).GT.-EPS)GO TO 120
         RATIO=-A(MC,J)/A(LEAVE,J)
         IF(RATIO.GE.RMIN)GO TO 120
         RMIN=RATIO
         ENTER=J
  120    CONTINUE
         STATUS(BVAR(LEAVE))=-1
         STATUS(ENTER)=0
         BVAR(LEAVE)=ENTER
         FACTOR=A(LEAVE,ENTER)
         DO 130 J=1,NB
C
C
C
```

```
  130  A(LEAVE,J)=A(LEAVE,J)/FACTOR
       DO 150 I=1,MC
       IF(I.EQ.LEAVE)GO TO 150
       X=A(I,ENTER)
       DO 140 J=1,NB
  140  A(I,J)=A(I,J)-X*A(LEAVE,J)
  150  CONTINUE
       GO TO 120
  160  XY=-A(MC,NB)
       WRITE(22,170)XY
  170  FORMAT(5X,'FIRST LP SOLUTION = ',F10.3)
       LBOBJ=XY+0.9999
       ICI=0
       TOP=0
       KOUNT=1
       OBOBJ=XLARGE
  180  CALL LOCMIN(ICI,OBOBJ,TIX,AMOUNT)
C      WRITE(22,181)(TIX(J),J=1,N)
 181   FORMAT(2X,'TIX(I) = ',8I4)
       XY=-A(MC,NB)
C      WRITE(22,191)AMOUNT
 191   FORMAT(2X,'AMOUNT = ',I4)
       IF(AMOUNT.NE.0)GO TO 210
       IF(ICI.GE.OBOBJ)GO TO 290
       OBOBJ=ICI
C      WRITE(22,193)OBOBJ
 193   FORMAT(2X,'OBOBJ = ',I10)
       DO 190 I=1,N
  190  IX(I)=TIX(I)
  200  IF((LBOBJ+ALPHA).GE.OBOBJ)GO TO 290
       IF(XY.GT.(OBOBJ+EPS-1))GO TO 230
  210  CALL SORROW(SV,SR)
C      WRITE(22,211)SR,SV
 211   FORMAT(2X,'SR = ',I4,'SV = ',I4)
       TOP=TOP+1
       STACK(TOP,1)=1
       STACK(TOP,2)=SV
       STACK(TOP,3)=A(SR,NB)
       UBL=STACK(TOP,3)
       LEAVE=SR
C      WRITE(22,213)UBL,SV,LEAVE
 213   FORMAT(2X,'UBL = ',I4,' SV = ',I4,'  LEAVE = ',I4)
       CALL UBOUND(LEAVE,UBL,FEAS)
C      WRITE(22,216)A(MC,NB)
 216   FORMAT(2X,'A(MC,NB) = ',F10.3)
       KOUNT=KOUNT+1
       IF(KOUNT.GE.100)GO TO 290
       IF(FEAS.EQ.0)GO TO 230
       XY=-A(MC,NB)
C      WRITE(22,220),OBOBJ,LBOBJ,XY
  220  FORMAT(1X,'OBOBJ = ',I8,'LBOBJ = ',I5,' OBJ = ',F9.3)
       IF(XY.GT.(OBOBJ+EPS-1))GO TO 230
       GO TO 180
  230  CALL SDBACK
       XY=-A(MC,NB)
       IF(XY.GT.(OBOBJ+EPS-1))GO TO 240
       IF(STACK(TOP,1).EQ.1)GO TO 250
  240  TOP=TOP-1
       IF(TOP.EQ.0)GO TO 290
       GO TO 230
  250  SV=STACK(TOP,2)
       STACK(TOP,1)=-1
       STACK(TOP,3)=STACK(TOP,3)+1
       DO 260 I=1,M
  260  IF(BVAR(I).EQ.SV)GO TO 270
       I=0
  270  LEAVE=I
       LBL=STACK(TOP,3)
       CALL LBOUND(LEAVE,LBL,FEAS)
       KOUNT=KOUNT+1
       IF(KOUNT.GE.100)GO TO 290
       IF(FEAS.EQ.0)GO TO 230
       XY=-A(MC,NB)
C      WRITE(22,*),KOUNT,TOP
C      WRITE(22,280)OBOBJ,LBOBJ,XY
  280  FORMAT(1X,'OBOBJ = ',I8,'LBOBJ = ',I5,' OBJ = ',F9.3)
       IF(XY.GT.(OBOBJ+EPS-1))GO TO 230
       GO TO 180
  290  CALL RTIME(N2)
       IRTIME=N2-N1
       WRITE(22,300)IRTIME
  300  FORMAT(5X,'CPU TIME = ',I10,' milli seconds')
       WRITE(22,310)OBOBJ
  310  FORMAT(5X,'THE OBJ VALUE =',I8)
C      WRITE(22,1320)(I,IX(I),I=1,N)
 1320 FORMAT(5X,'X-',I2,'=',I4)
       WRITE(22,320)KOUNT
  320  FORMAT(5X,'NO. OF NODES EXAMINED = ',I4)
```

```
      330   FORMAT(///)
      340   CONTINUE
            STOP
            END
```

```
C
C*************************************************************************
C                 THIS SUBROUTINE GENERATES THE DATA REQUIRED FOR
C                       THE CYCLIC STAFF SCHEDULING PROBLEM
C*************************************************************************
      SUBROUTINE GENRAT(ISEED,LOW,HIGH,CYCLE)
      INTEGER CYCLE(100),DIF,O
      INTEGER BVAR(100),DEMAND(100),ENTER,STATUS(200),TOP,SR,UBOBJ,OB
      INTEGER STACK(300,3),IX(100),SV,HIGH,FEAS,BVAR1(100)
      INTEGER GA(100,100),IX1(100),TC,STPOSN(100),SDEM(100),USDEM(100
     1AMOUNT,CL,PIX(100),USDEM1(100)
      REAL A(101,200),RX(200),COST(100),UB(200),LB(200),A1(101,200)
      COMMON A,BVAR,STATUS,UB,LB,MA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
     1GM,STPOSN,SDEM,USDEM,CL,DEMAND
      DO 10 I=1,M
      DO 10 J=1,M
   10 GA(I,J)=0
      DIF=M/M
      DO 20 J=1,M
      K=1
      DO 20 I=1,CL
      L=I+DIF*(J-1)
      IF(L.GT.M)L=L-M
      GA(L,J)=CYCLE(K)
   20 K=K+1
      DO 30 J=1,M
   30 STPOSN(J)=DIF*(J-1)+1
      CALL SETRAN(ISEED)
      DO 40 I=1,M
      U1=RAN(X)
      IDEM=LOW+(HIGH-LOW)*U1+0.5
      DEMAND(I)=IDEM
      A(I,NB)=-IDEM
   40 CONTINUE
      O=0
      DO 50 J=1,M
      L=J+1
      IF(J.EQ.M)L=1
      IF((GA(1,J).EQ.0).AND.(GA(1,L).EQ.1))O=O+1
   50 CONTINUE
      K=0
      DO 60 I=1,CL
   60 IF(CYCLE(I).EQ.1)K=K+1
      WRITE(22,70)O,K
   70 FORMAT(5X,'O = ',I2,'  K = ',I2)
      RETURN
      END
```

```
C
C
C*******************************************************************************
C                 THE FOLLOWING SUBROUTINE ROUNDS-OFF THE FIRST LP
C                     SOLUTION AND SOLVES SECOND LP IF NEEDED.
C*****************************************************************************
      SUBROUTINE INCUMB(TC1,UBOBJ,TIX,AMOUNT)
      INTEGER BVAR(100),DEMAND(100),ENTER,STATUS(200),TOP,SR,UBOBJ,OB
      INTEGER STACK(300,3),IX(100),SV,HIGH,FEAS,BVAR1(100)
      INTEGER GA(100,100),IX1(100),TC,STPOSN(100),SDEM(100),USDEM(100
     1AMOUNT,CL,TIX(100),USDEM1(100)
      REAL A(101,200),RX(200),COST(100),UB(200),LB(200),A1(101,200)
      COMMON A,BVAR,STATUS,UB,LB,NA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
     1GA,STPOSN,SDEM,USDEM,CL,DEMAND
      DO 30 I=1,N
      TIX(I)=0
      IF(STATUS(I))10,30,20
   10 RX(I)=LB(I)
      GO TO 30
   20 RX(I)=UB(I)
   30 CONTINUE
      DO 40 I=1,M
   40 RX(BVAR(I))=A(I,NB)
      CALL ROUND(RX,IX1,TIX,TC,AMOUNT,USDEM1)
      DO 50 I=1,N
   50 TIX(I)=IX1(I)
      TC1=TC
      IF(AMOUNT.EQ.0)RETURN
      DO 60 I=1,M
   60 USDEM(I)=USDEM1(I)
      IF((UBOBJ-TC1).LE.1)RETURN
      M1=0
      DO 80 I=1,M
      IF(USDEM(I).LE.0)GO TO 80
      M1=M1+1
      A1(M1,NB)=-USDEM1(I)
      DO 70 J=1,N
   70 A1(M1,J)=-GA(I,J)
   80 CONTINUE
      DO 90 J=1,N
   90 A1(M1+1,J)=1
      A1(M1+1,NB)=0
      CALL NEXTLP(M1,A1,RX,IX1,TIX,TC,AMOUNT,USDEM1)
      IF(AMOUNT.NE.0)RETURN
      DO 100 J=1,N
  100 TIX(J)=TIX(J)+IX1(J)
      TC1=TC1+TC
      RETURN
      END
```

```
C
C*************************************************************
C       THIS SUBROUTINE SOLVES ONE LP
C*************************************************************
        SUBROUTINE NEWLP(M1,A1,RX,IX1,TIX,TC,AMOUNT,USDEM1)
        INTEGER BVAR(100),DEMAND(100),ENTER,STATUS(200),TOP,SR,OBOBJ,OB
        INTEGER STACK(300,3),IX(100),SV,HIGH,FEAS,BVAR1(100)
        INTEGER GA(100,100),IX1(100),TC,STPOSN(100),SDEM(100),USDEM(100
        1AR)JNF,CL,PLY(100),USDEM1(100)
        REAL A(101,200),RX(200),COST(100),OB(200),GB(200),
        1A1(101,200)
        COMMON A,BVAR,STATUS,OB,GB,OA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
        1GA,STPOSN,SDEM,USDEM,CL,DEMAND
        IFER=0
        DO 20 J=N+1,N+M1
        DO 10 I=1,M1
        A1(I,J)=0
   10   IF((J-N).EQ.I)A1(I,J)=1
        A1(M1+1,J)=0
   20   BVAR1(J-N)=J
        A1(M1+1,NB)=0
   30   IFER=IFER+1
        BMIN=-EPS
        DO 40 I=1,M1
        IF(A1(I,NB).GE.BMIN)GO TO 40
        LEAVE=I
        BMIN=A1(I,NB)
   40   CONTINUE
        IF(BMIN.EQ.-EPS)GO TO 100
        RMIN=XLARGE
        DO 50 J=1,N+M1
        IF(A1(LEAVE,J).GE.-EPS)GO TO 50
        RATIO=-A1(M1+1,J)/A1(LEAVE,J)
        IF(RATIO.GE.RMIN)GO TO 50
        RMIN=RATIO
        ENTER=J
   50   CONTINUE
   60   FORMAT(' BMIN = ',F12.2,'  ENTER = ',I4)
        BVAR1(LEAVE)=ENTER
        FACTOR=A1(LEAVE,ENTER)
        DO 70 J=1,NB
   70   A1(LEAVE,J)=A1(LEAVE,J)/FACTOR
        DO 90 I=1,M1+1
        IF(I.EQ.LEAVE)GO TO 90
        X=A1(I,ENTER)
        DO 80 J=1,NB
   80   A1(I,J)=A1(I,J)-X*A1(LEAVE,J)
   90   CONTINUE
        GO TO 30
  100   DO 110 J=1,N
  110   RX(J)=0
        DO 120 I=1,M1
  120   RX(BVAR1(I))=A1(I,NB)
        CALL RDUAL(RX,IX1,TIX,TC,AMOUNT,USDEM1)
        RETURN
        END
```

```
C
C***********************************************************************
C        THIS SUBROUTINE DO THE SENSITIVITY ANALYSIS TO IMPOSE
C              UPPER BOUND CONSTRAINT ON ANY VARIABLE
C***********************************************************************
      SUBROUTINE UBOUND(LEAVE,UBL,FEAS)
      INTEGER ENTER,BVAR(100),STATUS(200),OLEAVE,STACK(300,3),TOP,FEA
      INTEGER GA(100,100),IXI(100),TC,STPOSN(100),SDEM(100),USDEM(100
     1ADJNT,DEMAND(100),CL,TIX(100),USDEM1(100)
      REAL A(101,200),UB(200),LB(200),LEMMIN,LEMDA
      COMMON A,BVAR,STATUS,UB,LB,NA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
     1GA,STPOSN,SDEM,USDEM,CL,DEMAND
      I1=STACK(TOP,2)
      OLEAVE=LEAVE
      FEAS=1
      KOUNT=0
      UB(I1)=A(LEAVE,NB)
      XI1=A(LEAVE,NB)
      C=1
   10 RMIN=XLARGE
      KOUNT=KOUNT+1
      DO 20 J=1,NA
      IF(LB(J).EQ.UB(J))GO TO 20
      XX=C*STATUS(J)*A(LEAVE,J)
      IF(XX.GE.-EPS)GO TO 20
      RATIO=C*A(MC,J)/A(LEAVE,J)
      IF(RATIO.GE.RMIN)GO TO 20
      RMIN=RATIO
      ENTER=J
   20 CONTINUE
      IF(RMIN.GE.XLARGE)GO TO 50
      FEAS=0
      IF(KOUNT.GT.1)GO TO 40
      IF(OLEAVE.EQ.0)GO TO 40
      UB(I1)=XLARGE
      IF(TOP.EQ.1)GO TO 40
      DO 30 J=1,(TOP-1)
      I=TOP-J
      IF(STACK(I,2).NE.I1)GO TO 30
      IF(STACK(I,1).EQ.-1)GO TO 30
      UB(I1)=STACK(I,3)
      GO TO 40
   30 CONTINUE
   40 RETURN
   50 IL=BVAR(LEAVE)
      FACTOR=A(LEAVE,ENTER)
      DO 60 J=1,NA
   60 A(LEAVE,J)=A(LEAVE,J)/FACTOR
      DO 80 I=1,MC
      IF(I.EQ.LEAVE)GO TO 80
      TEMP=A(I,ENTER)
      DO 70 J=1,NA
   70 A(I,J)=A(I,J)-A(LEAVE,J)*TEMP
   80 CONTINUE
      STATUS(IL)=-1
      IF(A(LEAVE,NB).GT.(LB(IL)+EPS))STATUS(IL)=1
      A(LEAVE,NB)=LB(ENTER)
      IF(STATUS(ENTER).EQ.1)A(LEAVE,NB)=UB(ENTER)
      BVAR(LEAVE)=ENTER
      STATUS(ENTER)=0
      LEMMIN=XLARGE
      DO 110 I=1,M
      LEMDA=XLARGE
      AC=A(I,I1)
      IF(ABS(AC).LE.EPS)GO TO 110
      IF(AC.GT.EPS)GO TO 90
      LEMDA=-(A(I,NB)-LB(BVAR(I)))/AC
      C1=-1
      GO TO 100
   90 UBI=UB(BVAR(I))
      IF(UBI.EQ.XLARGE)GO TO 100
      LEMDA=(UBI-A(I,NB))/AC
      C1=1
  100 IF(LEMDA.GE.LEMMIN)GO TO 110
      LEMMIN=LEMDA
      LEMROW=I
      C=C1
  110 CONTINUE
      CHANGE=XI1-UBL
      IF(LEMMIN.GT.CHANGE)LEMMIN=CHANGE
      DO 120 I=1,MC
  120 A(I,NB)=A(I,NB)+A(I,I1)*LEMMIN
      XI1=XI1-LEMMIN
      UB(I1)=XI1
      IF(CHANGE.EQ.LEMMIN)GO TO 130
      LEAVE=LEMROW
C
C
```

```
C
      130   GO TO 10
            RETURN
            END
```

```
C
C*****************************************************************************
C        THIS SUBROUTINE DO THE SENSITIVITY ANALYSIS TO IMPOSE
C            TO IMPOSE LOWER BOUND CONSTRAINT ON A VARIABLE
C*****************************************************************************
      SUBROUTINE LBOUND(LEAVE,LBL,FEAS)
      INTEGER ENTER,BVAR(100),STATUS(200),OLEAVE,STACK(300,3),TOP,FEA
      REAL A(101,200),UB(200),LB(200),LEMMIN,LEMDA
      INTEGER SA(100,100),IXI(100),TC,STPOSN(100),SDEM(100),USDEM(100
     1ASOJVT,DEMAND(100),CL,TIX(100),USDEM1(100)
      COMMON A,BVAR,STATUS,UB,LB,NA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
     1SA,STPOSN,SDEM,USDEM,CL,DEMAND
      I1=STACK(TOP,2)
      OLEAVE=LEAVE
      FEAS=1
      KOUNT=0
      IF(LEAVE.NE.0)GO TO 10
      XI1=LB(I1);IF(STATUS(I1).EQ.1)XI1=UB(I1)
      IF(XI1.GE.LBL)RETURN
      STATUS(I1)=-1
      GO TO 100
   10 LB(I1)=A(LEAVE,NB)
      XI1=A(LEAVE,NB)
      C=-1
   20 RMIN=XLARGE
      KOUNT=KOUNT+1
      DO 30 J=1,NA
      XX=C*STATUS(J)*A(LEAVE,J)
      IF(XX.GE.-EPS)GO TO 30
      IF(LB(J).EQ.UB(J))GO TO 30
      RATIO=C*A(MC,J)/A(LEAVE,J)
      IF(RATIO.GE.RMIN)GO TO 30
      RMIN=RATIO
      ENTER=J
   30 CONTINUE
      IF(RMIN.NE.XLARGE)GO TO 60
      FEAS=0
      IF(KOUNT.GT.1)GO TO 50
      IF(OLEAVE.EQ.0)GO TO 50
      LB(I1)=0
      IF(TOP.EQ.1)GO TO 50
      DO 40 J=1,(TOP-1)
      I=TOP-J
      IF(STACK(I,2).NE.I1)GO TO 40
      IF(STACK(I,1).EQ.1)GO TO 40
      LB(I1)=STACK(I,3)
      GO TO 50
   40 CONTINUE
   50 RETURN
   60 IL=BVAR(LEAVE)
      FACTOR=A(LEAVE,ENTER)
      DO 70 J=1,NA
   70 A(LEAVE,J)=A(LEAVE,J)/FACTOR
      DO 90 I=1,MC
      IF(I.EQ.LEAVE)GO TO 90
      TEMP=A(I,ENTER)
      DO 80 J=1,NA
   80 A(I,J)=A(I,J)-A(LEAVE,J)*TEMP
   90 CONTINUE
      STATUS(IL)=-1
      IF(A(LEAVE,NB).GT.(UB(IL)+EPS))STATUS(IL)=1
      A(LEAVE,NB)=LB(ENTER)
      IF(STATUS(ENTER).EQ.1)A(LEAVE,NB)=UB(ENTER)
      BVAR(LEAVE)=ENTER
      STATUS(ENTER)=0
  100 LEMMIN=XLARGE
      DO 130 I=1,M
      LEMDA=XLARGE
      AC=A(I,I1)
      IF(ABS(AC).LE.EPS)GO TO 130
      IF(AC.LT.-EPS)GO TO 110
      LEMDA=(A(I,NB)-LB(BVAR(I)))/AC
      C1=-1
      GO TO 120
  110 UBI=UB(BVAR(I))
      IF(UBI.EQ.XLARGE)GO TO 120
      LEMDA=-(UBI-A(I,NB))/AC
      C1=1
  120 IF(LEMDA.GE.LEMMIN)GO TO 130
      LEMMIN=LEMDA
      LROW=I
      C=C1
  130 CONTINUE
      CHANGE=LBL-XI1
      IF(LEMMIN.GT.CHANGE)LEMMIN=CHANGE
C
C
C
```

```
      DO 140 I=1,MC
140   A(I,NB)=A(I,NB)-A(I,II)*FACTM
      XII=XII+DELTA
      GB(II)=XII
      IF(CHANGE.GT.DELMIN)GO TO 150
      ESAVE=ERROR
      GO TO 20
150   DO 160 I=1,MC
160   CONTINUE
      RETURN
      END
```

```
C
C*****************************************************************
C          THIS SUBROUTINE DOES THE SENSITIVITY ANALYSIS TO RELAX
C                    A CONSTRAINT ON ANY VARIABLE.
C*****************************************************************
      SUBROUTINE GOBACK
      INTEGER ENTER,BVAR(100),STATUS(200),STACK(300,3),TOP,STATIR,C1
      INTEGER SA(100,100),IX1(100),TC,STPOSN(100),SDEM(100),USDEM(100
      IANDJNT,DEMAND(100),CL,FLAG,TIX(100),USDEM1(100)
      REAL A(101,200),UB(200),LB(200),LEMMIN,LEMDA
      COMMON A,BVAR,STATUS,UB,LB,MA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
     1SS,STPOSN,SDEM,USDEM,CL,DEMAND
      IR=STACK(TOP,2)
      STATIR=STATUS(IR)
      FLAG=1
      PVIR=LB(IR)
      IF(STATIR.EQ.1)PVIR=UB(IR)
      K2=STACK(TOP,1)
      IF((K2*STATIR).LE.0)FLAG=0
      IF(FLAG.EQ.1)GO TO 20
      IF(K2.EQ.1)UB(IR)=XLARGE
      IF(K2.EQ.-1)LB(IR)=0
      IF(TOP.EQ.1)RETURN
      DO 10 J=1,(TOP-1)
      I=TOP-J
      IF(STACK(I,2).NE.IR)GO TO 10
      IF(STACK(I,1).NE.K2)GO TO 10
      IF(K2.EQ.-1)LB(IR)=STACK(I,3)
      IF(K2.EQ.1)UB(IR)=STACK(I,3)
      RETURN
   10 CONTINUE
      RETURN
   20 STATUS(IR)=-STATIR
      IF(STATIR.EQ.-1)GO TO 40
      LB(IR)=UB(IR)
      UB(IR)=XLARGE
      IF(TOP.EQ.1)GO TO 60
      DO 30 J=1,(TOP-1)
      I=TOP-J
      IF(STACK(I,2).NE.IR)GO TO 30
      IF(STACK(I,1).EQ.-1)GO TO 30
      UB(IR)=STACK(I,3)
      GO TO 50
   30 CONTINUE
      GO TO 50
   40 UB(IR)=LB(IR)
      LB(IR)=0
      IF(TOP.EQ.1)GO TO 60
      DO 50 J=1,(TOP-1)
      I=TOP-J
      IF(STACK(I,2).NE.IR)GO TO 50
      IF(STACK(I,1).EQ.1)GO TO 50
      LB(IR)=STACK(I,3)
      GO TO 50
   50 CONTINUE
   60 DO 70 J=1,NA
      IF(LB(J).EQ.UB(J))GO TO 70
      XX=STATUS(J)*A(MC,J)
      IF(XX.GT.EPS)GO TO 80
   70 CONTINUE
      RETURN
   80 ENTER=J
      C1=STATUS(J)
      LEMMIN=XLARGE
      DO 110 I=1,M
      XX=C1*A(I,ENTER)
      IF(ABS(XX).LE.EPS)GO TO 110
      IF(XX.GT.0.)GO TO 90
      LEMDA=-C1*(A(I,NB)-UB(BVAR(I)))/A(I,ENTER)
      GO TO 100
   90 LEMDA=-C1*(A(I,NB)-LB(BVAR(I)))/A(I,ENTER)
  100 IF(LEMDA.GE.LEMMIN)GO TO 110
      LEMROW=I
      LEMMIN=LEMDA
  110 CONTINUE
      ID=BVAR(LEMROW)
      XENTER=LB(ENTER)
      IF(C1.EQ.1)XENTER=UB(ENTER)
      X2=UB(ENTER)-LB(ENTER)
      IF(LEMMIN.GE.X2)GO TO 120
      LEMMIN=X2
      STATUS(ENTER)=-C1
  120 DO 130 I=1,MC
  130 A(I,NB)=A(I,NB)+C1*A(I,ENTER)*LEMMIN
      IF(LEMMIN.EQ.X2)GO TO 170
      STATUS(ID)=-1
C
C
```

```
      C
            IF(A(LEAROW,NB).GE.(OB(IL)+EPS))STATUS(IL)=1
            A(LEMROW,NB)=XENTER-C1*LEMAIN
            BVAR(LEAROW)=ENTER
            STATUS(ENTER)=N
            FACTOR=A(LEMROW,ENTER)
            DO 140 J=1,NA
      140   A(LEMROW,J)=A(LEMROW,J)/FACTOR
            DO 160 I=1,MC
            IF(I.EQ.LEMROW)GO TO 160
            TEMP=A(I,ENTER)
            DO 150 J=1,NA
      150   A(I,J)=A(I,J)-A(LEMROW,J)*TEMP
      160   CONTINUE
      170   IF(ENTER.NE.IR)GO TO 210
            IF(STATIR.EQ.-1)GO TO 190
            OB(IR)=0
            IF(IOP.EQ.1)GO TO 210
            DO 180 I=1,(IOP-1)
            J=IOP-I
            IF(STACK(I,2).NE.IR)GO TO 180
            IF(STACK(I,1).EQ.1)GO TO 180
            OB(IR)=STACK(I,3)
            GO TO 210
      180   CONTINUE
            GO TO 210
      190   OB(IR)=XLARGE
            IF(IOP.EQ.1)GO TO 210
            DO 200 I=1,(IOP-1)
            J=IOP-I
            IF(STACK(I,2).NE.IR)GO TO 200
            IF(STACK(I,1).EQ.-1)GO TO 200
            OB(IR)=STACK(I,3)
            GO TO 210
      200   CONTINUE
      210   GO TO 60
            END
```

```
C
C*****************************************************************
C          THIS SUBROUTINE ROUNDS-OFF THE CONTINUOUS VALUED SOLUTION
C                    IN ALL WAYS AND GIVES THE BEST SOLUTION
C*****************************************************************
      SUBROUTINE ROUND(RX,IX1,TIX,TC,AMOUNT,USDEM1)
      INTEGER SDEM(100),USDEM(100),IX1(100),IX2(100),AMOUNT,TAMUNT,TC
     1TCOST,STPOSN(100),DEMAND(100),TUSDEM(100),BVAR(100),STACK(300,
     2CL,TOP,STATUS(200),GA(100,100),TIX(100),USDEM1(100)
      REAL A(101,200),RX(200),COST(100),UB(200),LB(200)
      COMMON A,AVAR,STATUS,UB,LB,MA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
     1GA,STPOSN,SDEM,USDEM,CL,DEMAND
      AMOUNT=XLARGE
      DO 70 I1=1,N
      SX=0
      TCOST=0
      DO 10 J=I11,I11+N-1
      J1=J
      IF(J.GT.N)J1=J-N
      SX1=SX
      R2=RX(J1)
      K2=R2
      IF((R2-K2).LE.EPS)R2=K2
      IF((R2-K2).GE.(1-EPS))R2=K2+1
      SX=SX1+R2
      I1=SX1+(1.-EPS)
      I2=SX+(1.-EPS)
      IX2(J1)=I2-I1
   10 TCOST=TCOST+IX2(J1)
      DO 20 I=1,M
   20 SDEM(I)=0
      DO 40 J=1,N
      DO 30 I=STPOSN(J),STPOSN(J)+CL-1
      L=I
      IF(I.GT.M)L=I-M
   30 IF(GA(L,J).EQ.1)SDEM(L)=SDEM(L)+IX2(J)+TIX(J)
   40 CONTINUE
      TAMUNT=0
      DO 50 I=1,M
      TUSDEM(I)=DEMAND(I)-SDEM(I)
      IF(TUSDEM(I).GT.0)TAMUNT=TAMUNT+TUSDEM(I)
   50 CONTINUE
      IF(TAMUNT.GE.AMOUNT)GO TO 70
      AMOUNT=TAMUNT
      TC=TCOST
      DO 60 I=1,M
      USDEM1(I)=TUSDEM(I)
      IX1(I)=IX2(I)
   60 IF(AMOUNT.EQ.0)GO TO 80
   70 CONTINUE
   80 RETURN
      END
```

```
C
C*****************************************************************
C          THIS SUBROUTINE FINDS THE SOURCE VARIABLE FOR BRANCHING
C*****************************************************************
      SUBROUTINE SOURCE(SV,SR)
      INTEGER BVAR(100),STATUS(200),STACK(300,3),TOP,SR,SV
      INTEGER GA(100,100),IX1(100),TC,STPOSN(100),SDEM(100),USDEM(100
     1)
      REAL DEMAND(100),CL,TIX(100),USDEM1(100)
      REAL A(101,201),UB(200),LB(200)
      COMMON A,BVAR,STATUS,UB,LB,MA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
     1GA,STPOSN,SDEM,USDEM,CL,DEMAND
      TAMT=-EPS
      SV=0
      SR=0
      DO 20 J=1,M
      IBV=BVAR(J)
      IF(IBV.GT.0)GO TO 20
C     IF(A(J,NB).LT.(IX1(IBV)+EPS))GO TO 20
      R1=A(J,NB)
      I1=R1+EPS
      I2=R1+(1.-EPS)
      IF(I1.EQ.I2)GO TO 20
      K1=0
      ISP=STPOSN(J)
      DO 10 III=ISP,ISP+CL-1
      I=III
      IF(I.GT.M)I=I-M
   10 IF((GA(I,J).EQ.1).AND.(USDEM(I).GT.0))K1=K1+1
      FRAC=I2-R1
      AMT=K1*FRAC
      IF(AMT.LT.TAMT)GO TO 20
      TAMT=AMT
      SR=J
      SV=IBV
   20 CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C          DUAL PRIMAL CUTTING PLANE ALGORITHM
C*****************************************************************
      DOUBLE PRECISION SUMCOL
      REAL A(0:200,1:101),BA(100,100),RX(200),ARRFU(200,2)
      REAL ARRFL(200,0:51),MAXRAT
      INTEGER M,N,R,IX(100),POSN(200),COUNT,HIGH,DEMAND(100)
      INTEGER SEED(100),P,SR,CYCLE(100),CL,ASJUNT
      LOGICAL ITER,FEAS
      COMMON A,M,N
      READ(23,*)NPROB
      DO 1100 I1=1,NPROB
      WRITE(24,14)I1
14    FORMAT(1X,'PROBLEM -',I2)
      WRITE(24,17)
17    FORMAT(5X,'CUTTING PLANE METHOD')
      READ(23,*)M,N,CL
      MO=M
      WRITE(24,30)M,N
30    FORMAT(5X,'M = ',I3,'  N = ',I3)
      DO 40 J=1,N
      A(0,J)=0
40    IF(J.GE.N)A(0,J)=1
      READ(23,*)ISEED,LOW,HIGH
      WRITE(24,50)ISEED,LOW,HIGH
50    FORMAT(5X,'SEED = ',I3,'   RANGE OF DEMANDS IS ',I3,' to ',I3)
      READ(23,*)(CYCLE(I),I=1,CL)
      WRITE(24,122)(CYCLE(I),I=1,CL)
122   FORMAT(5X,'CYCLE:  ',3612)
      CALL GENRAT(ISEED,LOW,HIGH,CYCLE,CL,DEMAND)
      CALL RTEMP(N2)
      M=M+N+1
      DO 120 J=1,N
120   A(M-N,J)=-1
      A(0,N+1)=0
      A(M-N,N+1)=0
      DO 160 I1=M-N+1,M
      DO 150 J1=1,N+1
160   A(I1,J1)=0.0
      DO 180 I1=M-N+1,M
      A(I1,I1-M+N)=-1
180   CONTINUE
      DO 200 I=1,M
      IF(I.GE.N)POSN(I)=M-N+1
      IF(I.GT.N)POSN(I)=I-N
200   CONTINUE
      DO 220 I=1,M-N-1
      DO 220 J=1,N
220   BA(I,J)=A(I,J)
      EPS=0.0001
      NCUTS=0
      NCUTS=0
      MOO=M
280   NP=N+1
300   R=0
320   R=R+1
      ITER=A(R,NP) < -EPS
      IF((.NOT.ITER).AND.(R.LT.M))GO TO 320
      IF(.NOT.ITER) GO TO 540
      COUNT=COUNT+1
      ITR=R
      DO 340 N1=ITR+1,M
      IF(A(R,NP).GT.A(N1,NP))R=N1
340   CONTINUE
360   K=0
380   K=K+1
      ITER=A(R,K) < -EPS
      IF((.NOT.ITER).AND.(K.LT.N)) GO TO 380
      FEAS=ITER
      IF(.NOT.ITER) GO TO 540
      L=K
      IF(K.EQ.N)GO TO 440
      DO 420 J=K+1,N
      IF(A(R,J).GE.-EPS) GO TO 420
      I=-1
400   I=I+1
      S=A(I,J)/A(R,J)-A(I,L)/A(R,L)
      IF(S.EQ.0.0) GO TO 400
      IF(S.GT.0.0)L=J
420   CONTINUE
440   TEMP1=A(R,L)
460   DO 480 I=0,M
480   A(I,L)=-A(I,L)/TEMP1
      DO 520 J=1,NP
      IF(J.EQ.L) GO TO 520
      TEMP=A(R,J)
      DO 500 I=0,M
500   A(I,J)=A(I,J)+A(I,L)*TEMP
520   CONTINUE
```

```
540   IF(ITER)GO TO 300
      SUMX=-A(0,N+1)
      IF((NOCUTS.EQ.0).AND.(NCUTS.EQ.0))FLPSOL=SUMX
      TYPE *,NOCUTS,NCUTS,SUMX
      ISUMX=SUMX+EPS
      RISUMX=ISUMX
      IF((RISUMX+EPS).GT.SUMX)GO TO 600
      DO 580 J=1,N
      SUMCOL=0.
      DO 560 I=(MG-N+1),MG
560   SUMCOL=SUMCOL+A(I,J)
580   A(MG-N,J)=SUMCOL
      A(MG-N,N+1)=-(RISUMX+1.)-A(0,N+1)
      NCUTS=NCUTS+1
      GO TO 280
600   DO 620 I=MG-N+1,MG
620   RX(I-MG+N)=A(I,N+1)
      A(0,N+1)=-RISUMX
      DO 1000 III=1,N
      SX=0.
      DO 640 J1=III,III+N
      J=J1
      IF(J1.GT.MD)J=J1-MD
      SX1=SX
      SX=SX1+RX(J)
      I1=SX1+0.999
      I2=SX+0.999
640   IX(J)=I2-I1
      AMOUNT=0
      DO 680 I=1,MG-N-1
      SDEM(I)=0
      DO 660 J=1,N
660   IF(JA(I,J).EQ.-1.0)SDEM(I)=SDEM(I)+IX(J)
      USDEM=DEMAND(I)-SDEM(I)
      IF(USDEM.GT.0.0)AMOUNT=AMOUNT+1
680   CONTINUE
      IF(AMOUNT.EQ.0)GO TO 900
1000  CONTINUE
      DO 700 I=1,MG
      KK=POSN(I)
      IP=A(KK,N+1)+EPS
      IF((IP+EPS).LT.(A(KK,N+1))) GO TO 720
      A(KK,N+1)=IP
700   CONTINUE
      GO TO 900
720   M=M+1
      POSN(M)=M
      P=1
      ARRFO(P,1)=KK
      ARRFO(P,2)=A(KK,N+1)-IP
      DO 740 L1=1,M
      LL=POSN(L1)
      IP1=A(LL,N+1)+EPS
      IF(IP1.GT.A(LL,N+1))A(LL,N+1)=IP1
      P=P+1
      ARRFO(P,2)=A(LL,N+1)-IP1
      ARRFO(P,1)=LL
740   CONTINUE
      DO 760 K=1,P-1
      DO 760 I=K,P-1
      IF(ARRFO(K,2).GE.ARRFO(I+1,2))GO TO 760
      TEMP1=ARRFO(K,1)
      TEMP2=ARRFO(K,2)
      ARRFO(K,1)=ARRFO(I+1,1)
      ARRFO(K,2)=ARRFO(I+1,2)
      ARRFO(I+1,1)=TEMP1
      ARRFO(I+1,2)=TEMP2
760   CONTINUE
      DO 860 I=1,20
      KK=ARRFO(I,1)
      SUMFIJ=0
      ARRFIJ(I,N+1)=ARRFO(I,2)
      NZFIJ=0
      DO 840 J=1,N
      IP=A(KK,J)
      IF(A(KK,J).LT.-EPS)IP=IP-1
      FIJ=A(KK,J)-IP
      IF(FIJ.GE.EPS)GO TO 780
      FIJ=0
      M1=A(KK,J)
      A(KK,J)=M1
780   IF(FIJ.GT.(1.0-EPS))GO TO 800
      GO TO 820
800   FIJ=0.
      A(KK,J)=IP+1.
820   ARRFIJ(I,J)=FIJ
      SUMFIJ=SUMFIJ+FIJ
      IF(FIJ.NE.0.)NZFIJ=NZFIJ+1
840   CONTINUE
```

```
      ARRFIJ(I,J)=RC
      ARRFIJ(I,N+2)=SLKFIJ
      IF(SLKFIJ.GT.0.)GO TO 860
      ARFIJ=ARRFIJ(I,J+1)/SLKFIJ*AZFIJ
      IF(ARFIJ.LE.MAXRAT)GO TO 860
      MAXRAT=ARFIJ
      SR=I
  860 CONTINUE
      DO 880 J=1,N+1
  880 A(0,J)=-ARRFIJ(SR,J)
      NCUTS=NCUTS+1
      IF(NCUTS.GT.150)GO TO 986
      GO TO 480
  986 WRITE(21,989)
  989 FORMAT(5X,'THE PROBLEM COULD NOT BE SOLVED AFTER 150 CUTS')
  900 WRITE(24,920)NCUTS
  920 FORMAT(5X,'NO. OF FRACTIONAL CUTS = ',I4)
      WRITE(24,940)NCUTS
  940 FORMAT(5X,'NO. OF CUTS =',I4)
C
      WRITE(24,960)(I,IX(I),I=1,N)
  960 FORMAT(5X,'X-',I2,I4)
      ITC=-(A(0,N+1)-EPS)
      WRITE(24,980)ITC
  980 FORMAT(5X,'TOTAL COST = ',I5)
      WRITE(24,984)FLPSOL
  984 FORMAT(5X,'FIRST LP SOLUTION = ',F10.3)
      CALL PTIME(T3)
      ICPUT=T3-T2
      WRITE(24,126)ICPUT
  126 FORMAT(5X,'TOTAL CPU TIME = ',I8,' milli seconds')
      WRITE(24,995)
  995 FORMAT(////)
 1100 CONTINUE
      STOP
      END
      SUBROUTINE GENRAT(ISEED,LOW,HIGH,CYCLE,CL,DEMAND)
      REAL A(0:200,1:101)
      INTEGER CYCLE(100),DIF,HIGH,DEMAND(100),CL
      COMMON A,M,N
      DO 10 I=1,M
      DO 10 J=1,N
   10 A(I,J)=0
      DIF=M/N
      DO 20 J=1,N
      K=1
      DO 20 I=1,CL
      L=1+DIF*(J-1)
      IF(L.GT.M)L=L-M
      A(L,J)=-CYCLE(K)
   20 K=K+1
      CALL SETRAN(ISEED)
      DO 40 I=1,M
      U1=RAN(X)
      IDEM=LOW+(HIGH-LOW)*U1+0.5
      DEMAND(I)=IDEM
      A(I,N+1)=-IDEM
   40 CONTINUE
      RETURN
      END
```

```
C***********************************************************************
C        BARTHOLDI'S METHOD
C***********************************************************************
      INTEGER BVAR(100),ENTER,IX(200),DEMAND(100),HIGH
      INTEGER CPUT,CYCLE(100),CL,Q
      REAL A(0:100,0:200),RX(200)
      [illegible]
      READ(21,*)NPROB
      DO 250 IP=1,NPROB
      WRITE(22,5)IP
 5    FORMAT(1X,'PROBLEM -',I2)
      WRITE(22,10)
 10   FORMAT(5X,'BARTHOLDI''S METHOD')
      READ(21,*)M,N,CL
      DO 20 J=1,N
      A(0,J)=0
 20   IF(J.LE.M)A(0,J)=1
      READ(21,*)ISEED,LOW,HIGH
      READ(21,*)(CYCLE(I),I=1,CL)
      WRITE(22,45)(CYCLE(I),I=1,CL)
 45   FORMAT(5X,'CYCLE: ',36I2)
      CALL GENRAT(ISEED,LOW,HIGH,CYCLE,CL,DEMAND)
      A(0,J)=0.
      TCOST=0.
      Q=0
      DO 30 J=1,N
      L=J+1
      IF(J.EQ.N)L=1
      IF((A(1,J).EQ.0).AND.(A(1,L).EQ.-1.))Q=Q+1
 30   CONTINUE
      DO 40 I=1,M
 40   A(I,NB)=A(I,NB)-Q+1
      CALL RTIME(NI)
      NA=N+M
      NB=0
      BIGM=100000
      EPS=0.00001
      ITER=0
      DO 60 J=N+1,NA
      DO 50 I=1,M
      A(I,J)=0
      IF((J-N).EQ.I)A(I,J)=1
 50   CONTINUE
      A(0,J)=0
      BVAR(J-N)=J
 60   CONTINUE
      A(0,NB)=0
 70   ITER=ITER+1
      BMIN=-EPS
      DO 80 I=1,M
      IF(A(I,NB).GT.BMIN)GO TO 80
      LEAVE=I
      BMIN=A(I,NB)
 80   CONTINUE
      IF(BMIN.EQ.-EPS)GO TO 130
      RMIN=BIGM
      DO 90 J=1,NA
      IF(A(LEAVE,J).GT.-EPS)GO TO 90
      RATIO=-A(0,J)/A(LEAVE,J)
      IF(RATIO.GE.RMIN)GO TO 90
      RMIN=RATIO
      ENTER=J
 90   CONTINUE
      BVAR(LEAVE)=ENTER
      FACTOR=A(LEAVE,ENTER)
      DO 100 J=0,NA
 100  A(LEAVE,J)=A(LEAVE,J)/FACTOR
      DO 120 I=0,M
      IF(I.EQ.LEAVE)GO TO 120
      X=A(I,ENTER)
      DO 110 J=0,NA
 110  A(I,J)=A(I,J)-X*A(LEAVE,J)
 120  CONTINUE
      GO TO 70
 130  CONTINUE
      XX=-A(0,NB)
      DO 140 I=1,NA
 140  RX(I)=0
      DO 150 I=1,M
 150  RX(BVAR(I))=A(I,NB)
      SX=0
      DO 155 J=1,N
      SX1=SX
      SX=SX+RX(J)
      I1=SX1+0.99999
C
C
C
C
```

```
      I2=SX+0.99999
      IX(J)=I2-I1
      ICOST=ICOST+IX(J)
160   CONTINUE
      CALL TIMER(M2)
      K=0
      DO 170 I=1,CL
170   IF(CYCLE(I).EQ.1)K=K+1
      BOUND=N*(M-1)/K
      IBOUND=BOUND+0.9999
      CPUT=M2-M1
      WRITE(22,180)M,N
180   FORMAT(5X,'M = ',I4,' N = ',I4)
      WRITE(22,190)ISEED,LOW,HIGH
190   FORMAT(5X,'SEED = ',I5,'      RANGE OF DEMANDS IS ',I4,' to ',I3)
      WRITE(22,200)TCOST
200   FORMAT(5X,'TOTAL COST =',F8.2)
      WRITE(22,210)Q,K
210   FORMAT(5X,'Q = ',I2,'  K = ',I3)
      WRITE(22,220)IBOUND
220   FORMAT(5X,'THE BOUND FOR THIS SOLUTION IS ',I6)
      WRITE(22,230)CPUT
230   FORMAT(5X,'TOTAL CPU TIME TAKEN = ',I5,' milli seconds')
      WRITE(22,240)
240   FORMAT(////)
250   CONTINUE
      STOP
      END
      SUBROUTINE GENRAT(ISEED,LOW,HIGH,CYCLE,CL,DEMAND)
      REAL A(0:100,0:200)
      INTEGER CYCLE(100),DIF,HIGH,DEMAND(100),CL
      COMMON A,M,N
      DO 10 I=1,M
      DO 10 J=1,N
10    A(I,J)=0.
      DIF=M/Q
      DO 20 J=1,N
      K=1
      DO 20 I=1,CL
      L=I+DIF*(J-1)
      IF(L.GT.M)L=L-M
      A(L,J)=-CYCLE(K)
20    K=K+1
      CALL SETRAN(ISEED)
      DO 30 I=1,M
      U1=RAN(X)
      IDEM=LOW+(HIGH-LOW)*U1+0.5
      DEMAND(I)=IDEM
      A(I,J)=-IDEM
30    CONTINUE
      RETURN
      END
```